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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 January 2016 (14.01.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/007832 A1

(51) International Patent Classification:
H04N 19/70 (2014.01) *H04N 19/593* (2014.01)
H04N 19/90 (2014.01)

(21) International Application Number:
PCT/US2015/039904

(22) International Filing Date:
10 July 2015 (10.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

62/023,436	11 July 2014 (11.07.2014)	US
14/795,734	9 July 2015 (09.07.2015)	US

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; ATTN: International IP Administration, 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventors: **PU, Wei**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **ZOU, Feng**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **KARCZEWICZ, Marta**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **JOSHI, Rajan Laxman**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **SEREGIN, Vadim**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **SOLE ROJALS, Joel**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

[Continued on next page]

(54) Title: ADVANCED PALETTE PREDICTION AND SIGNALING

(57) Abstract: A video coder may determine a palette predictor list comprising one or more candidates. Each respective candidate in the palette predictor list specifies a value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data. The video coder may include, in a palette for the current block, at least one candidate in the palette predictor list.

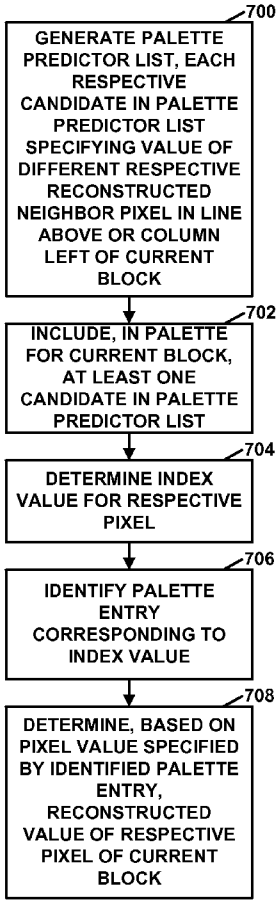

FIG. 7

WO 2016/007832 A1

(74) **Agent**: **VREDEVELD, Albert W.**; Shumaker & Sieffert, P.A., 1625 Radio Drive, Suite 300, Woodbury, Minnesota 55125 (US).

(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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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))*

ADVANCED PALETTE PREDICTION AND SIGNALING

[0001] The application claims the benefit of U.S. Provisional Patent Application 62/023,436, filed July 11, 2014, the entire content of which is incorporated by reference.

TECHNICAL FIELD

[0002] This disclosure relates to video encoding and video decoding.

BACKGROUND

[0003] Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called "smart phones," video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video compression techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), the High Efficiency Video Coding (HEVC) standard, and extensions of such standards. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video compression techniques.

[0004] Video compression techniques perform spatial (intra-picture) prediction and/or temporal (inter-picture) prediction to reduce or remove redundancy inherent in video sequences. For block-based video coding, a video slice (i.e., a video frame or a portion of a video frame) may be partitioned into video blocks. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

[0005] Spatial or temporal prediction results in a predictive block for a block to be coded. Residual data represents pixel differences between the original block to be coded and the predictive block. An inter-coded block is encoded according to a motion vector

that points to a block of reference samples forming the predictive block, and the residual data indicates the difference between the coded block and the predictive block. An intra-coded block is encoded according to an intra-coding mode and the residual data. For further compression, the residual data may be transformed from the pixel domain to a transform domain, resulting in residual coefficients, which then may be quantized. The quantized coefficients, initially arranged in a two-dimensional array, may be scanned in order to produce a one-dimensional vector of coefficients, and entropy coding may be applied to achieve even more compression.

SUMMARY

[0006] In general, this disclosure describes techniques for palette mode coding of video data. As described herein, a video coder may determine a palette predictor list that includes one or more candidate entries. Each of the candidate entries may specify at least one sample value of a different, respective reconstructed pixel neighboring a current block. The video coder may include one or more candidate entries of the palette predictor list in a palette of the current block.

[0007] In one example, this disclosure describes a method of decoding video data, the method comprising: determining a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is adjacent to a current block of the video data; including, in a palette for the current block, at least one candidate in the palette predictor list; obtaining, based on one or more syntax elements signaled in a bitstream, an index value for a pixel of the current block; identifying a palette entry in the palette that corresponds to the index value for the pixel; and determining, based on the at least one sample value specified by the identified palette entry, at least one reconstructed sample value of the pixel.

[0008] In another example, this disclosure describes a method of encoding video data, the method comprising: determining a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of

the video data; including, in a palette for the current block, at least one candidate in the palette predictor list; and signaling, in a bitstream, one or more syntax elements indicating an index value for a pixel of the current block, the index value corresponding to an entry in the palette.

[0009] In another example, this disclosure describes a device configured to code video data, the device comprising: a memory configured to store the video data; and one or more processors configured to: determine a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data; and include, in a palette for the current block, at least one candidate in the palette predictor list.

[0010] In another example, this disclosure describes a device configured to decode video data, the device comprising: means for determining a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data; and means for including, in a palette for the current block, at least one candidate in the palette predictor list.

[0011] In another example, this disclosure describes a data storage medium having instructions stored thereon that when executed cause a device for coding video data to: determine a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data; and include, in a palette for the current block, at least one candidate in the palette predictor list.

[0012] The details of one or more examples of the disclosure are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description, drawings, and claims.

BRIEF DESCRIPTION OF DRAWINGS

[0013] FIG. 1 is a block diagram illustrating an example video coding system that may utilize the techniques described in this disclosure.

[0014] FIG. 2 is a conceptual diagram illustrating an example palette stuffing process.

[0015] FIG. 3 is a conceptual diagram illustrating example prediction flags.

[0016] FIG. 4 is a conceptual diagram illustrating candidate pixels for an extra palette predictor list.

[0017] FIG. 5 is a block diagram illustrating an example video encoder that may implement the techniques described in this disclosure.

[0018] FIG. 6 is a block diagram illustrating an example video decoder that may implement the techniques described in this disclosure.

[0019] FIG. 7 is a flowchart illustrating an example decoding operation, in accordance with a technique of this disclosure.

[0020] FIG. 8 is a flowchart illustrating an example encoding operation, in accordance with a technique of this disclosure.

[0021] FIG. 9 is a flowchart illustrating an example operation for generating a palette, in accordance with a technique of this disclosure.

[0022] FIG. 10 is a flowchart illustrating an example operation of a video decoder to generate an original palette predictor list, in accordance with a technique of this disclosure.

[0023] FIG. 11 is a flowchart illustrating an example operation of a video decoder to generate an extra palette predictor list, in accordance with a technique of this disclosure.

[0024] FIG. 12 is a flowchart illustrating an example operation of a video encoder to generate an original palette predictor list, in accordance with a technique of this disclosure.

[0025] FIG. 13 is a flowchart illustrating an example operation of a video encoder to generate an extra palette predictor list, in accordance with a technique of this disclosure.

DETAILED DESCRIPTION

[0026] Aspects of this disclosure are directed to techniques for video coding and video data compression. In particular, this disclosure describes techniques for palette-based coding of video data. In traditional video coding, images are assumed to be continuous-tone and spatially smooth. Based on these assumptions, various tools have been

developed such as block-based transform, filtering, and other coding tools and such tools have shown good performance for natural content videos.

[0027] However, in applications like remote desktop, collaborative work and wireless display, computer generated screen content may be the dominant content to be compressed. This type of content tends to have discrete tones, sharp lines, and high contrast object boundaries. The assumption of continuous-tone and smoothness may no longer apply, and thus, traditional video coding techniques may be inefficient in compressing the content.

[0028] This disclosure describes palette-based coding, which may be particularly suitable for screen content coding or other content where one or more traditional coding tools are inefficient. Palette-based coding of video data may be used with one or more other coding techniques, such as techniques for inter- or intra-predictive coding. For example, as described in greater detail below, an encoder or decoder, or combined encoder-decoder (codec), may be configured to perform inter- and intra-predictive coding, as well as palette-based coding.

[0029] In palette mode coding, a palette includes or consists of entries numbered by an index representing color component values, which can be used as predictor for block samples or as final reconstructed block samples. Each entry in the palette may contain one color component (for example, a luma value) or two components (for example, two chroma values) or a triplet of three color components (for example, RGB, YUV, or components of other color spaces). Four or more color components might also be used in other situations, such as with four-component color spaces like "CMYK."

[0030] A block of video data may be coded by indicating, for each sample of the block, an index to a corresponding entry in the palette. Further efficiency is achieved by indicating runs of indexes to the same entry in the palette instead of individually signaling indexes. Samples in the palette mode block may be coded using run-length coding with run-modes, e.g., 'copy-left' (which may also be referred to herein as 'index-copy'), and 'copy-above.'

[0031] To reduce bits used to signal a palette, entries in the palette are predicted using previously decoded palette entries. The previous decoded palette entries (after pruning out duplicate entries) form a list, referred to as a palette predictor list. For each entry in the palette predictor list, one bin is used to indicate whether the corresponding entry is re-used in the current palette or not. These bins may form a binary vector. This vector corresponds to the syntax elements of previous_palette_entry_flag[i].

[0032] In the decoder, after decoding all of the previous_palette_entry_flag, the entries in the palette predictor list whose corresponding previous_palette_entry_flag equals to 1 are placed at the beginning of the current block's palette. In other words, the current palette can be logically divided into two zones. The first zone (named as ZONE_P hereafter, placed at the beginning) is composed of re-used entries from the palette predictor list. The second zone (named as ZONE_T hereafter, placed after the first zone) is composed of new palette entries, which are not in the palette predictor list and directly signaled in the bitstream.

[0033] The palette predictor list does not include so-called "escape pixels" used in a previously-coded block. Escape pixels are pixels present in a palette-coded block, but not included in the palette for the block. There may be an elevated probability the escape pixels of the previously coded block are also used in the current block, especially if those escape pixels are adjacent to the current block. Hence, it may be advantageous to include at least some of the escape pixels of the previously-coded block in the palette predictor list for the current block.

[0034] This disclosure describes techniques using neighboring reconstructed pixels, such as pixels from the line above a current block and the line to the left of the current block as additional candidate palette predictors to predict the current block's palette. For example, a video coder may use neighboring reconstructed pixels to construct another palette prediction list, referred to herein as the "extra palette predictor list." A video coder may use pixels in the "extra palette predictor list" in the same way the video coder would use other pixels in a palette. For instance, the bitstream may include an index to an entry in the "extra palette predictor list" to indicate that a corresponding pixel of a current block has the sample value specified by the entry.

[0035] Therefore, in accordance with an example of this disclosure, a video coder may determine a palette predictor list comprising or consisting of one or more candidates. In this example, each respective candidate in the palette predictor list specifies at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Furthermore, in this example, each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data. For example, the neighboring pixels may be adjacent to the current block. In some examples, the neighboring pixels may come from the above row or two or more rows above the current block. The neighboring pixels may also come from the left column or left two or more columns of the current block. In this example,

the video coder includes, in a palette for the current block, at least one candidate in the palette predictor list.

[0036] FIG. 1 is a block diagram illustrating an example video coding system 10 that may utilize the techniques of this disclosure. As used herein, the term "video coder" refers generically to both video encoders and video decoders. In this disclosure, the terms "video coding" or "coding" may refer generically to video encoding or video decoding.

[0037] As shown in FIG. 1, video coding system 10 includes a source device 12 and a destination device 14. Source device 12 generates encoded video data. Accordingly, source device 12 may be referred to as a video encoding device or a video encoding apparatus. Destination device 14 may decode the encoded video data generated by source device 12. Accordingly, destination device 14 may be referred to as a video decoding device or a video decoding apparatus. Source device 12 and destination device 14 may be examples of video coding devices or video coding apparatuses. Decoding denotes reconstructing sample values from encoded video data.

[0038] Source device 12 and destination device 14 may comprise a wide range of devices, including desktop computers, mobile computing devices, notebook (e.g., laptop) computers, tablet computers, set-top boxes, telephone handsets such as so-called "smart" phones, televisions, cameras, display devices, digital media players, video gaming consoles, in-car computers, video conferencing devices, or other types of media-enabled devices.

[0039] Destination device 14 may receive encoded video data from source device 12 via a channel 16. Channel 16 may comprise one or more media or devices capable of moving the encoded video data from source device 12 to destination device 14. In one example, channel 16 may comprise one or more communication media that enable source device 12 to transmit encoded video data directly to destination device 14 in real-time. In this example, source device 12 may modulate the encoded video data according to a communication standard, such as a wireless communication protocol, and may transmit the modulated video data to destination device 14. The one or more communication media may include wireless and/or wired communication media, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The one or more communication media may form part of a packet-based network, such as a local area network, a wide-area network, or a global network (e.g., the Internet). The one or

more communication media may include routers, switches, base stations, or other equipment that facilitate communication from source device 12 to destination device 14.

[0040] In another example, channel 16 may include a storage medium that stores encoded video data generated by source device 12. In this example, destination device 14 may access the storage medium via disk access or card access. The storage medium may include a variety of locally-accessed data storage media such as Blu-ray discs, DVDs, CD-ROMs, flash memory, or other suitable digital storage media for storing encoded video data.

[0041] In a further example, channel 16 may include a file server or another intermediate storage device that stores encoded video data generated by source device 12. In this example, destination device 14 may access encoded video data stored at the file server or other intermediate storage device via streaming or download. The file server may be a type of server capable of storing encoded video data and transmitting the encoded video data to destination device 14. Example file servers include web servers (e.g., for a website), file transfer protocol (FTP) servers, network attached storage (NAS) devices, and local disk drives.

[0042] Destination device 14 may access the encoded video data through a standard data connection, such as an Internet connection. Example types of data connections may include wireless channels (e.g., Wi-Fi connections), wired connections (e.g., DSL, cable modem, etc.), or combinations of both that are suitable for accessing encoded video data stored on a file server. The transmission of encoded video data from the file server may be a streaming transmission, a download transmission, or a combination of both.

[0043] The techniques of this disclosure are not limited to wireless applications or settings. The techniques may be applied to video coding in support of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, streaming video transmissions, e.g., via the Internet, encoding of video data for storage on a data storage medium, decoding of video data stored on a data storage medium, or other applications. In some examples, video coding system 10 may be configured to support one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

[0044] FIG. 1 is merely an example and the techniques of this disclosure 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 (e.g., video data) is retrieved from a local memory, streamed over a network, or the like. A video encoding device may encode and store data (e.g., video data) to memory, and/or a video decoding device may retrieve and decode data (e.g., video data) from memory. In many examples, the encoding and decoding is performed by devices that do not communicate with one another, but simply encode data (e.g., video data) to memory and/or retrieve and decode data (e.g., video data) from memory.

[0045] In the example of FIG. 1, source device 12 includes a video source 18, a video encoder 20, and an output interface 22. In some examples, output interface 22 may include a modulator/demodulator (modem) and/or a transmitter. Video source 18 may include a video capture device, e.g., a video camera, a video archive containing previously-captured video data, a video feed interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources of video data.

[0046] Video encoder 20 may encode video data from video source 18. In some examples, source device 12 directly transmits the encoded video data to destination device 14 via output interface 22. In other examples, the encoded video data may also be stored onto a storage medium or a file server for later access by destination device 14 for decoding and/or playback.

[0047] In the example of FIG. 1, destination device 14 includes an input interface 28, a video decoder 30, and a display device 32. In some examples, input interface 28 includes a receiver and/or a modem. Input interface 28 may receive encoded video data from channel 16. Display device 32 may be integrated with or may be external to destination device 14. In general, display device 32 displays decoded video data. Display device 32 may comprise a variety of display devices, such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0048] This disclosure may generally refer to video encoder 20 "signaling" or "transmitting" certain information to another device, such as video decoder 30. The term "signaling" or "transmitting" may generally refer to the communication of syntax elements and/or other data used to decode the compressed video data. Such communication may occur in real- or near-real-time. Alternately, such communication may occur over a span of time, such as might occur when storing syntax elements to a computer-readable storage medium in an encoded bitstream at the time of encoding, which then may be retrieved by a decoding device at any time after being stored to this

medium. Thus, while video decoder 30 may be referred to as "receiving" certain information, the receiving of information does not necessarily occur in real- or near-real-time and may be retrieved from a medium at some time after storage.

[0049] Video encoder 20 and video decoder 30 each may be implemented as any of a variety of suitable circuitry, such as one or more microprocessors, digital signal processors (DSPs), application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), discrete logic, hardware, or any combinations thereof. 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. Any of the foregoing (including hardware, software, a combination of hardware and software, etc.) may be considered to be one or more processors. Each of video encoder 20 and video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device.

[0050] In some examples, video encoder 20 and video decoder 30 operate according to a video compression standard, such as HEVC. A version of the HEVC, referred to as "HEVC Version 1" hereinafter, is available from https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.265-201304-S!!PDF-E&type=items. In addition to HEVC Version 1, there are ongoing efforts to produce scalable video coding, multiview video coding, and 3D coding extensions for HEVC. In addition, palette-based coding modes, e.g., as described in this disclosure, may be provided for extension of HEVC. In some examples, the techniques described in this disclosure for palette-based coding may be applied to encoders and decoders configured to operate according to other video coding standards, such as the ITU-T-H.264/AVC standard or future standards. Accordingly, application of a palette-based coding mode for coding of coding units (CUs) or prediction units (PUs) in an HEVC codec is described for purposes of example.

[0051] Video encoder 20 and video decoder 30 of video coding system 10 represent examples of devices that may be configured to perform techniques for palette-based video coding in accordance with various examples described in this disclosure. For example, video encoder 20 and video decoder 30 may be configured to selectively code various blocks of video data, such as CUs or PUs in HEVC coding, using either palette-based coding or non-palette based coding. Non-palette based coding modes may refer to

various inter-predictive temporal coding modes or intra-predictive spatial coding modes, such as the various coding modes specified by HEVC Version 1.

[0052] In HEVC and other video coding standards, a video sequence typically includes a series of pictures. Pictures may also be referred to as "frames." A picture may include three sample arrays, denoted S_L, S_{Cb} and S_{Cr}. S_L is a two-dimensional array (i.e., a block) of luma samples. S_{Cb} is a two-dimensional array of Cb chrominance samples. S_{Cr} is a two-dimensional array of Cr chrominance samples. Chrominance samples may also be referred to herein as "chroma" samples. In other instances, a picture may be monochrome and may only include an array of luma samples.

[0053] To generate an encoded representation of a picture, video encoder 20 may generate a set of coding tree units (CTUs). Each of the CTUs may be a coding tree block of luma samples, two corresponding coding tree blocks of chroma samples, and syntax structures used to code the samples of the coding tree blocks. A coding tree block may be an NxN block of samples. A CTU may also be referred to as a "tree block" or a "largest coding unit" (LCU). The CTUs of HEVC may be broadly analogous to the macroblocks of other standards, such as H.264/AVC. However, a CTU is not necessarily limited to a particular size and may include one or more coding units (CUs). A slice may include an integer number of CTUs ordered consecutively in the raster scan.

[0054] To generate a coded CTU, video encoder 20 may recursively perform quad-tree partitioning on the coding tree blocks of a CTU to divide the coding tree blocks into coding blocks, hence the name "coding tree units." A coding block is an NxN block of samples. A CU may be a coding block of luma samples and two corresponding coding blocks of chroma samples of a picture that has a luma sample array, a Cb sample array and a Cr sample array, and syntax structures used to code the samples of the coding blocks. In monochrome pictures or pictures having three separate color planes, a CU may comprise a single coding block and syntax structures used to code the samples of the coding block.

[0055] Video encoder 20 may partition a coding block of a CU into one or more prediction blocks. A prediction block may be a rectangular (i.e., square or non-square) block of samples on which the same prediction is applied. A prediction unit (PU) of a CU may be a prediction block of luma samples, two corresponding prediction blocks of chroma samples of a picture, and syntax structures used to predict the prediction block samples. Video encoder 20 may generate predictive luma, Cb and Cr blocks for luma,

Cb and Cr prediction blocks of each PU of the CU. In monochrome pictures or pictures having three separate color planes, a PU may comprise a single prediction block and syntax structures used to predict the prediction block.

[0056] Video encoder 20 may use intra prediction or inter prediction to generate the predictive blocks for a PU. Predictive blocks may also be referred to as "predictive sample blocks." If video encoder 20 uses intra prediction to generate the predictive blocks of a PU, video encoder 20 may generate the predictive blocks of the PU based on decoded samples of the picture associated with the PU. In this disclosure, "based on" may, in some instances, be interpreted as "based at least in part on."

[0057] If video encoder 20 uses inter prediction to generate the predictive blocks of a PU, video encoder 20 may generate the predictive blocks of the PU based on decoded samples of one or more pictures other than the picture associated with the PU. Video encoder 20 may use uni-prediction or bi-prediction to generate the predictive blocks of a PU. When video encoder 20 uses uni-prediction to generate the predictive blocks for a PU, the PU may have a single motion vector. When video encoder 20 uses bi-prediction to generate the predictive blocks for a PU, the PU may have two motion vectors.

[0058] After video encoder 20 generates predictive blocks (e.g., predictive luma, Cb and Cr blocks) for one or more PUs of a CU, video encoder 20 may generate a residual block for the CU. Each sample in the residual block indicates a difference between a sample in one of the CU's predictive blocks and a corresponding sample in one of the CU's original coding blocks. For example, video encoder 20 may generate a luma residual block for the CU. Each sample in the CU's luma residual block indicates a difference between a luma sample in one of the CU's predictive luma blocks and a corresponding sample in the CU's original luma coding block. In addition, video encoder 20 may generate a Cb residual block for the CU. Each sample in the CU's Cb residual block may indicate a difference between a Cb sample in one of the CU's predictive Cb blocks and a corresponding sample in the CU's original Cb coding block. Video encoder 20 may also generate a Cr residual block for the CU. Each sample in the CU's Cr residual block may indicate a difference between a Cr sample in one of the CU's predictive Cr blocks and a corresponding sample in the CU's original Cr coding block.

[0059] Furthermore, video encoder 20 may use quad-tree partitioning to decompose the residual blocks (e.g., luma, Cb and Cr residual blocks) of a CU into one or more transform blocks (e.g., luma, Cb and Cr transform blocks). A transform block may be a

rectangular block of samples on which the same transform is applied. A transform unit (TU) of a CU may be a transform block of luma samples, two corresponding transform blocks of chroma samples, and syntax structures used to transform the transform block samples. Thus, each TU of a CU may correspond to a luma transform block, a Cb transform block, and a Cr transform block. The luma transform block associated with the TU may be a sub-block of the CU's luma residual block. The Cb transform block may be a sub-block of the CU's Cb residual block. The Cr transform block may be a sub-block of the CU's Cr residual block. In monochrome pictures or pictures having three separate color planes, a TU may comprise a single transform block and syntax structures used to transform the samples of the transform block.

[0060] Video encoder 20 may apply one or more transforms to a transform block to generate a coefficient block for the TU. A coefficient block may be a two-dimensional array of transform coefficients. A transform coefficient may be a scalar quantity. For example, video encoder 20 may apply one or more transforms to a luma transform block of a TU to generate a luma coefficient block for the TU. Video encoder 20 may apply one or more transforms to a Cb transform block of a TU to generate a Cb coefficient block for the TU. Video encoder 20 may apply one or more transforms to a Cr transform block of a TU to generate a Cr coefficient block for the TU.

[0061] After generating a coefficient block (e.g., a luma coefficient block, a Cb coefficient block or a Cr coefficient block), video encoder 20 may quantize the coefficient block. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the transform coefficients, providing further compression. After video encoder 20 quantizes a coefficient block, video encoder 20 may entropy encode syntax elements indicating the quantized transform coefficients. For example, video encoder 20 may perform Context-Adaptive Binary Arithmetic Coding (CABAC) on the syntax elements indicating the quantized transform coefficients. Video encoder 20 may output the entropy-encoded syntax elements in a bitstream. The bitstream may also include syntax elements that are not entropy encoded.

[0062] Video encoder 20 may output a bitstream that includes the entropy-encoded syntax elements. The bitstream may include a sequence of bits that forms a representation of coded pictures and associated data. In other words, the bitstream may comprise an encoded representation of video data. The bitstream may comprise a sequence of network abstraction layer (NAL) units. Each of the NAL units includes a

NAL unit header and encapsulates a raw byte sequence payload (RBSP). The NAL unit header may include a syntax element indicating a NAL unit type code. The NAL unit type code specified by the NAL unit header of a NAL unit indicates the type of the NAL unit. A RBSP is a syntax structure containing an integer number of bytes that is encapsulated within a NAL unit. In some instances, an RBSP includes zero bits.

[0063] Different types of NAL units may encapsulate different types of RBSPs. For example, a first type of NAL unit may encapsulate an RBSP for a picture parameter set (PPS), a second type of NAL unit may encapsulate an RBSP for a coded slice, a third type of NAL unit may encapsulate an RBSP for Supplemental Enhancement Information (SEI), and so on. NAL units encapsulating RBSPs for video coding data (as opposed to RBSPs for parameter sets and SEI messages) may be referred to as video coding layer (VCL) NAL units. SEI may contain (e.g., consist of) information that is not necessary to decode the samples of coded pictures from VCL NAL units. An SEI RBSP may contain one or more SEI messages.

[0064] A VPS is a syntax structure comprising syntax elements applying to zero or more entire coded video sequences (CVSs). A sequence parameter set (SPS) is also a syntax structure comprising syntax elements applying to zero or more entire CVSs. An SPS may include a syntax element identifying a VPS active when the SPS is active. Thus, the syntax elements of a VPS may be more generally applicable than the syntax elements of an SPS. A picture parameter set (PPS) is a syntax structure comprising syntax elements applying to zero or more coded pictures. A PPS may include a syntax element identifying an SPS active when the PPS is active. A slice header of a slice may include a syntax element that indicates a PPS active when the slice is being coded.

[0065] Video decoder 30 may receive a bitstream generated by video encoder 20. In addition, video decoder 30 may parse the bitstream to obtain (e.g., decode) syntax elements from the bitstream. Video decoder 30 may reconstruct the pictures of the video data based at least in part on the syntax elements decoded from the bitstream. The process to reconstruct the video data may be generally reciprocal to the process performed by video encoder 20.

[0066] For instance, video decoder 30 may use motion vectors of PUs to determine predictive blocks for the PUs of a current CU. In addition, video decoder 30 may inverse quantize transform coefficient blocks associated with TUs of the current CU. Video decoder 30 may perform inverse transforms on the transform coefficient blocks to reconstruct transform blocks associated with the TUs of the current CU. Video

decoder 30 may reconstruct the coding blocks of the current CU by adding the samples of the predictive sample blocks for PUs of the current CU to corresponding samples of the transform blocks of the TUs of the current CU. By reconstructing the coding blocks for each CU of a picture, video decoder 30 may reconstruct the picture.

[0067] In some examples, video encoder 20 and video decoder 30 may be configured to perform palette-based coding. For example, in palette-based coding, video encoder 20 and video decoder 30 may code a so-called palette as a table of colors for representing the video data of the particular area (e.g., a given block). Each pixel may be associated with an entry in the palette that represents at least one sample of the pixel. For example, video encoder 20 and video decoder 30 may code an index for a pixel identifying a palette entry specifying one or more samples of the pixel. In some examples, a "pixel value" is a single sample value of a single color component (e.g., luma, Cb, Cr). In other examples, a "pixel value" is a plurality of sample values (e.g., a luma sample value, a Cb sample value, and a Cr sample value).

[0068] Video encoder 20 may signal palette-related syntax elements at a CU level. For instance, if a CU is encoded using palette mode, the CU may include one or more palette_coding_component syntax structures. A syntax of the palette_coding_component syntax structure is reproduced in Table 1, below.

TABLE 1

palette_coding_component(x0, y0, CbWidth, CbHeight, NumComp) {	**Descriptor**
compOffset = (NumComp == 3) ? 0 : (NumComp – 1)	
nCbS = (1 << log2CbSize)	
numPredPreviousPalette = 0	
for(i = 0; i < previousPaletteSize; i++) {	
previous_palette_entry_flag[i]	ae(v)
if (previous_palette_entry_flag[i]) {	
for (cIdx = compOffset; cIdx < NumComp + compOffset; cIdx++)	
palette_entries[cIdx][numPredPreviousPalette] = previousPaletteEntries[cIdx][i]	
numPredPreviousPalette++	
}	
}	
if(numPredPreviousPalette < max_palette_size)	
palette_num_signalled_entries	ae(v)
for (cIdx = compOffset; cIdx < NumComp + compOffset; cIdx++)	
for(i = 0; i < palette_num_signalled_entries; i++)	
palette_entries[cIdx][numPredPreviousPalette + i]	ae(v)
palette_size = numPredPreviousPalette + palette_num_signalled_entries	
previous_run_type_flag = INDEX_MODE	
scanPos = 0	
while(scanPos < nCbS * nCbS) {	
xC = scanPos % nCbS	
yC = scanPos / nCbS	
if (yC != 0 && previous_run_type_flag != COPY_ABOVE_MODE)	
palette_run_type_flag[xC][yC]	ae(v)
else	
palette_run_type_flag[xC][yC] = INDEX_MODE	

previous_run_type_flag = palette_run_type_flag[xC][yC]	
if(palette_run_type_flag[xC][yC] == INDEX_MODE) {	
palette_index	ae(v)
if(palette_index == palette_size) { /* ESCAPE_PIXEL */	
scanPos++	
for(cIdx = compOffset; cIdx < NumComp + compOffset; cIdx++) {	
palette_escape_val	ae(v)
samples_array[cIdx][xC][yC] = palette_escape_val	
}	
}	
}	
if(palette_run_type_flag[xC][yC] == COPY_ABOVE_MODE || (palette_run_type_flag[xC][yC] == INDEX_MODE && palette_index < palette_size)) {	
palette_run	ae(v)
runPos = 0	
while (runPos <= palette_run) {	
if(palette_run_type_flag[xC][yC] == INDEX_MODE)	
paletteMap[xC][yC] = palette_index	
else	
paletteMap[xC][yC] = paletteMap[xC][yC − 1]	
for(cIdx = compOffset; cIdx < NumComp + compOffset; cIdx++)	
samples_array[cIdx][xC][yC] = palette_entries[cIdx][paletteMap[xC][yC]]	
runPos++	
scanPos++	
xC = scanPos % nCbS	
yC = scanPos / nCbS	
}	

}	
}	
previousPaletteSize = palette_size	
for(i = 0; i < palette_size; i++)	
for (cIdx = compOffset; cIdx < NumComp + compOffset; cIdx++)	
previousPaletteEntries[cIdx][i] = palette_entries[cIdx][i]	
}	

[0069] The following paragraphs describe exemplary semantics of syntax elements in Table 1.

[0070] **previous_palette_entry_flag**[i] equal to 1 specifies that the palette entry from the previous used palette is copied. previous_palette_entry_flag [i] equals to 0 specifies that the palette entry from the previous used palette is not copied. The previousPaletteSize is the size of the palette of the previous CU decoded in palette mode. The variable previousPaletteSize is set equal to 0 for the first decoded palette CU of each CTB line as well as slice.

[0071] **palette_num_signalled_entries** specifies the number of entries in the palette that are explicitly signalled for the current CU. When palette_num_signalled_entries is not present, it is inferred to be equal to 0.

[0072] **palette_entries**[cIdx][i] specifies the i-th element in the palette for the color component cIdx. The array previousPaletteEntries is equal to the array palette_entries of the last CU decoded in palette mode.

[0073] **palette_escape_val** specifies the escape pixel value.

[0074] **palette_run_type_flag** equal to 1 specifies that the decoding process is COPY_ABOVE_MODE where the decoded pixel value is equal to the pixel at the same location in the above row. palette_run_type_flag equal to 0 specifies that the decoding process mode is INDEX_MODE where the pixel's palette index is coded in the bitstream. When palette_run_type_flag is not present, it is inferred to be equal to INDEX_MODE.

[0075] **palette_index** is an index to the palette entries. If palette_index is equal to palette_size, the pixel is coded as ESCAPE_PIXEL.

[0076] **palette_run** indicates the number of consecutive locations minus 1 with the same palette index as the position in the above row when palette_run_type_flag is equal to COPY_ABOVE_MODE or represents the number of consecutive locations minus 1 with the same palette index when palette_run_type_flag is equal to INDEX_MODE.

[0077] Thus, in palette mode (i.e., palette-based coding), a palette may consist of entries numbered by an index representing color component values which can be used as a predictor for block samples or as final reconstructed block samples. Each entry in the palette may contain one color component (for example, luma value) or two components (for example, two chroma values) or a triplet of three color components (for example, RGB, YUV etc.). Samples in the palette mode block are coded using run-length coding. For example, three run-modes may be used, including 'copy-left' (which may also be referred to herein as 'index-copy'), 'copy-above', and 'escape.' *See e.g.,* Pu et al., "SCCE3 base Software Bugfix Report," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 18th Meeting, Sapporo, JP, 30 June – 9 July 2014, document JCTVC-R0306 (hereinafter, "JCTVC-R0306-WD").

[0078] In some examples, video encoder 20 may encode a block of video data by determining a palette for the block, locating an entry in the palette to represent the value of each pixel, and encoding the palette with index values for the pixels relating the pixel value to the palette. Video decoder 30 may obtain, from an encoded bitstream, a palette for a block, as well as index values for the pixels of the block. Video decoder 30 may relate the index values of the pixels to entries of the palette to reconstruct the pixel values of the block.

[0079] To reduce bits used to signal a palette, entries in the palette may be predicted using previously decoded palette entries (*See e.g.,* JCTVC-R0306-WD *and* Gisquet et al., "SCCE3 Test A.3: palette stuffing," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 18th Meeting, Sapporo, JP, 30 June – 9 July 2014, document JCTVC-R0082 (hereinafter, "JCTVC-R0082"). The previously decoded palette entries, after pruning out duplicated entries, form a list, referred to herein as a palette predictor list. For each entry in the palette predictor list, one bin is used to indicate whether the corresponding entry is re-used in the current palette or not. These bins form a binary vector. In JCTVC-R0306-WD, this vector corresponds to the syntax elements previous_palette_entry_flag[i]. Various methods have been developed to compress this vector, such as Karczewicz et al.,

"SCCE3: Test A.8 – Improvements on palette prediction vector signaling," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 18th Meeting, Sapporo, JP, 30 June – 9 July 2014, document JCTVC-R0063 (hereinafter, "JCTVC-R0063").

[0080] Gisquet et al., "AhG10: palette predictor stuffing," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 18th Meeting, Sapporo, JP, 30 June – 9 July 2014, document JCTVC-Q0063 (hereinafter, "JCTVC-Q0063") provides another example of how to generate the palette predictor list. In JCTVC-Q0063, a video coder uses the last coded palette as a predictor. JCTVC-Q0063 proposed stuffing, at the end of this predictor, the unused values from the current predictor, as shown in FIG. 2. FIG. 2 is a conceptual diagram illustrating an example palette stuffing process. In FIG. 2, after CU1 has been encoded, a predictor is made available to CU2. Currently, it is the palette of CU1. Bit flags indicating the reuse of the predictor elements are generated and coded. CU2 palette would normally be used as the predictor of CU3. By checking the bit flags from the palette predictor of CU2, it is possible to determine which elements have not been reused. Those elements are added to the end of the predictor.

[0081] Including the additional elements in the predictor may be beneficial because those elements have actually occurred in the past, and may still occur. However, the probability of those additional elements occurring again is lower. By this recursive construction, a predictor is generated, where elements are somewhat ordered by probability. In addition, it may be possible and beneficial to increase the size of the predictor, due to the high compression ratio a valid prediction offers.

[0082] Particularly when a size of the predictor is much larger than the actual palette to predict, it may happen that the palette is predicted by elements mostly at the start of the predictor, as depicted is FIG. 3. FIG. 3 is a conceptual diagram illustrating example prediction flags. As can be seen in FIG. 3, the prediction scheme may have a limited efficiency. A typical improvement would be CABAC coding of the elements, but this would require frequent updating for at least 32 flags. JCTVC-Q0063 proposed instead to add intermediate flags, called "end-of-prediction," to indicate when elements can be no longer predicted. However, because this adds more flags, it may cause an expansion of the bitmap of flags. Therefore, it may be better to insert flags at specific locations in the bitmap.

[0083] In some instances, at video decoder 30, after decoding all of the previous_palette_entry_flag syntax elements, the entries in the palette predictor list whose corresponding previous_palette_entry_flag equals to 1 are placed at the beginning of the palette of the current block. In other words, the current palette can be logically divided into two zones. The first zone (named as **ZONE_P** hereafter, which is placed at the beginning) is composed of re-used entries from the palette predictor list. The second zone (named as **ZONE_T** hereafter, which is placed after the first zone) is composed of new palette entries which are not in the palette predictor list and directly signaled in the bitstream.

[0084] Palette predictor list does not include so-called "escape pixels" used in a previously-coded block. Escape pixels are pixels present in a palette-coded block, but not included in the palette for the block. For instance, escape pixels may be directly signaled and, in some examples, are not predictively coded. There may be an elevated probably the escape pixels of the previous block are also used in the current block, especially if those escape pixels are adjacent to the current block. Hence, it may be advantageous to include at least some of the escape pixels of the previously-coded block in the palette predictor list for the current block. If such escape pixels are not included in a palette predictor list, but are used in the current block, such pixels may need to be signaled as additional palette entries of the current block or escape pixels of the current block.

[0085] This disclosure proposes the use of neighboring reconstructed pixels (e.g., pixels from one or more lines immediately above the current block, if available, and one or more lines immediately to the left of the current block, if available) as additional candidate palette predictors to predict the current block's palette. To avoid confusion, this disclosure refers to the palette predictor list based on a palette of a previous block, such as the palette predictor list used in JCTVC-Q0063, as the "original palette predictor list."

[0086] This disclosure proposes the use of neighboring reconstructed pixels to construct another palette prediction list, referred to herein as an "extra palette predictor list." For example, if the neighboring reconstructed pixels specify sample values 3, 5, and 7, a video coder may determine an extra palette predictor list that includes candidates (i.e., entries) specifying sample values 3, 5, and 7.

[0087] For example, video encoder 20 may determine a palette predictor list consisting one or more candidates. In this example, each respective candidate in the palette

predictor list specifies at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. In this example, each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data. Furthermore, in this example, video encoder 20 may include, in a palette for the current block, at least one candidate in the palette predictor list. In this example, video encoder 20 may signal, in a bitstream, one or more syntax elements indicating an index value for a pixel of the current block, the index value corresponding to an entry in the palette.

[0088] In a similar example, video decoder 30 may determine a palette predictor list comprising or consisting of one or more candidates. In this example, each respective candidate in the palette predictor list specifies at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. In this example, each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data. Furthermore, in this example, video decoder 30 may include, in a palette for the current block, at least one candidate in the palette predictor list. In some instances, video decoder 30 does not include any of the candidates in the palette for the current block. Video decoder 30 may also obtain, based on one or more syntax elements signaled in a bitstream, an index value for a pixel of the current block. For example, the one or more syntax elements may specify the index value. In another example, video decoder 30 may obtain the index value by inferring, based on the one or more syntax elements and a particular rule, the index value for the pixel of the current block. In this example, video decoder 30 may identify a palette entry in the palette that corresponds to the index value for the pixel. Video decoder 30 may also determine, based on the at least one sample value specified by the identified palette entry, at least one reconstructed sample value of the pixel.

[0089] In accordance with some examples of this disclosure, assuming there are ***N*** (>= 0 or > 0) entries in the "extra palette predictor list," video encoder 20 may signal a binary vector (e.g., an array of binary values) whose size is equal to ***N*** in the bitstream to indicate whether each individual entry in the "extra palette predictor list" is re-used by the current block or not. For example, if ***N*** is equal to 5, video encoder 20 may signal a binary vector 11101 to indicate that the first, second, third, and fifth candidates in the extra palette predictor list are reused in the current block. If a candidate is not reused in the current block, a video coder may omit the candidate from the palette of the current

block. The binary vector may be transmitted directly in the bitstream or compressed using the method used to compress the original palette predictor list.

[0090] In this way, video encoder 20 may determine a palette predictor list that comprises one or more candidates. Each respective candidate in the palette predictor list may specify a value of a different reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block (e.g., CU, macroblock, etc.) of the video data. Furthermore, video encoder 20 may signal, in a bitstream, a binary vector comprising one or more syntax elements. Each of the one or more syntax elements corresponds to a different candidate in the palette predictor list. For each respective candidate in the palette predictor list, if the syntax element corresponding to the respective candidate indicates that a palette for a current block of the video data includes the value specified by the respective candidate, the palette includes a palette entry specifying the value specified by the respective candidate. If the syntax element corresponding to the respective candidate indicates that the palette does not include the value specified by the respective candidate, the palette omits the palette entry specifying the value specified by the respective candidate. Furthermore, video encoder 20 may signal, in the bitstream, one or more syntax elements indicating an index value for a sample of the current block, the index value corresponding to an entry in the palette. In some examples, the palette entries in the palette for the current block may specify residual pixel values. In other examples, the palette entries for the current block may specify non-residual pixel values.

[0091] Furthermore, in this way, video decoder 30 may determine a palette predictor list that comprises one or more candidates, each respective candidate in the palette predictor list specifying a value of a different reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block (e.g., CU, macroblock, etc.) of the video data. Video decoder 30 may obtain, from a bitstream, a binary vector comprising one or more syntax elements, wherein each of the one or more syntax elements corresponds to a different candidate in the palette predictor list. For each respective candidate in the palette predictor list, if the syntax element corresponding to the respective candidate indicates that a palette for a current block of the video data includes the value specified by the respective candidate, video decoder 30 may include, in the palette, a palette entry specifying the value specified by the

respective candidate. If the syntax element corresponding to the respective candidate indicates that the palette does not include the value specified by the respective candidate, video decoder 30 may omit, from the palette, the palette entry specifying the value specified by the respective candidate. Furthermore, video decoder 30 may obtain, based on one or more additional syntax elements signaled in a bitstream, an index value for a sample of the current block. Additionally, video decoder 30 may identify a palette entry in the palette that corresponds to the index value for the sample. Video decoder 30 may determine, based on a pixel value specified by the identified palette entry, a reconstructed value of the sample. For instance, in some examples, the reconstructed value of the sample may match the pixel value specified by the identified palette entry. In other example, the pixel value specified by the identified palette entry is a residual pixel value and video decoder 30 may determine the reconstructed value of the sample such that the reconstructed value of the sample is equal to a sum of the pixel value specified by the palette entry and a pixel value of a predictive block.

[0092] In accordance with one or more techniques of this disclosure, a video coder may perform an "extra palette predictor list" construction method. In a first example construction method, a video coder picks up to K (where $K > 0$) neighboring reconstructed pixels to form the "extra palette predictor list." In some examples, K=5 and the video coder uses the pixels A_0, A_1, B_0, B_1, and B_2, in FIG. 4. FIG. 4 is a conceptual diagram illustrating candidate pixels for an extra palette predictor list. For example, if pixels A_0, A_1, B_0, B_1, and B_2 specify sample values 5, 10, 15, 20, and 25, respectively, the video coder may determine an extra palette predictor list including candidates specifying sample values 5, 10, 15, 20, and 25, respectively. Thus, the reconstructed neighboring pixels used in the extra palette prediction list may include at least one of: a below-left reconstructed neighboring pixel (i.e., A_0), a left reconstructed neighboring pixel (i.e., A_1), an above-left reconstructed neighboring pixel (i.e., B_2), an above reconstructed neighboring pixel (i.e., B_1), or an above-right reconstructed neighboring pixel (i.e., B_0). An order is defined to rank the K candidates. As an example, the candidates may be ordered as (B_2, A_0, B_0, A_1, B_1). A pruning process may be applied to remove duplicate candidates. After pruning, if there are less than K candidates, certain predefined candidates may be added. For example, an entry with YUV value (0, 0, 0) may be added. As another example, an entry with YUV value (128, 128, 128) may be added.

[0093] The video coder may remove redundant candidates from the list. In some examples, a candidate close to another candidate before itself in the list may also be removed from the list. For instance, the video coder may remove a particular candidate from the extra palette predictor list if another candidate in the extra palette predictor list is close to the particular candidate. For example, if a threshold distance is 5, a sample value specified by a first candidate in the extra palette predictor list is 100, and a sample value specified by a second candidate in the extra palette predictor list is 99, the video coder may remove either the first or second candidates from the extra palette predictor list. The 'closeness' can be measured using metrics such as sum of absolute differences (SAD) or sum of squared error (SSE).

[0094] In another example construction method, a video coder ranks the pixels in the above line and left column or a subset of the pixels coming from the above line and left column according to their number of occurrence. High frequency pixels are placed in front of the low frequency pixels in the extra palette predictor list. Thus, in some examples, as part of determining the extra palette predictor list, the video coder may rank, within the extra palette predictor list, two or more of the plurality of reconstructed neighboring pixels according to frequency of sample values of the plurality of the reconstructed neighboring pixels. For example, if the extra palette predictor list includes candidates *A*, *B*, *C*, and *D*, and candidate *D* corresponds to a sample value occurring twice among the neighboring pixels, while candidates *A*, *B*, and *C* correspond to sample values only occurring once among the neighboring pixels, the video coder may rank, within the extra palette predictor list, the candidates such that candidate *D* precedes candidates *A*, *B*, and *C*.

[0095] This ranking may be based on the assumption that if a sample value is used more frequently among the neighboring pixels, the sample value is more likely to be used in the current block. Furthermore, because index values indicating palette entries may be represented as unary codes, associating more likely sample values with smaller index values may reduce bitstream size. Hence, ranking more likely candidates at the beginning of the extra palette predictor list may reduce bitstream size.

[0096] In some examples, the video coder may apply a list truncation process to cut the candidate list size to ***N*** if there are more than ***N*** candidates in the extra palette predictor list. Thus, in some examples, as part of determining the extra palette predictor list, the video coder may truncate the extra palette predictor list such that a size of the extra palette predictor list is limited to a particular number of candidates. For example, if ***N*** is

4 and the extra palette predictor list includes candidates *A*, *B*, *C*, *D*, *E*, and *F*, the video coder may truncate the extra palette predictor list such that the extra palette predictor list only includes candidates *A*, *B*, *C*, and *D*.

[0097] Furthermore, in some examples, the reconstructed neighboring pixels may be 'before in-loop filtering' values to reduce decoding delay. For instance, a video coder may apply in-loop filtering to the one or more reconstructed neighboring pixels after determining the palette predictor list.

[0098] In accordance with one or more techniques of this disclosure, after a video coder constructs the extra palette predictor list and the original palette predictor list for the current block, the video coder may apply a pruning process. The video coder may construct the extra palette predictor list on per block basis. For example, an entry in the extra palette predictor list is removed if the entry is also contained in original palette predictor list. Furthermore, in some examples, an entry in the extra palette predictor list is removed if the minimum distance between this entry and any entry in the original palette predictor list is within a threshold. For example, if the distance is 5, a sample value specified by a candidate in the original palette predictor list is 101, and a sample value specified by a candidate in the extra palette predictor list is 99, the video coder may remove either the candidate in the original palette predictor list or the candidate in the extra palette predictor list. The threshold may depend on quantization parameters. In some examples, the distance can be SAD, SSE, or other metrics.

[0099] In some examples, the entries in the extra palette predictor list may not be merged into the original palette predictor list after decoding the current block. Thus, in such examples, candidates of the palette predictor list are not included in a palette predictor list used to determine a palette for a block of the video data processed after the current block.

[0100] In accordance with one or more techniques of this disclosure, assume that the palette entries in the current block coming from an extra palette predictor list form a new zone named as **ZONE_E**. A video coder may order the three zones in accordance with one or more of the following examples. In a first example, the video coder puts the entries coming from the extra palette predictor list to the beginning of the current block's palette, then the palette entries coming from the original palette predictor list. In other words, in this first example, the three zones are ordered as **ZONE_E**, **ZONE_P**, **ZONE_T**. In a second example, the video coder puts the entries coming from the original palette predictor list to the beginning of the current block's palette, then the

palette entries coming from the extra palette predictor list. In other words, in this second example, the three zones are ordered as **ZONE_P**, **ZONE_E**, **ZONE_T**. Other orders, such as **ZONE_P**, **ZONE_T**, **ZONE_E**, may also be used.

[0101] FIG. 5 is a block diagram illustrating an example video encoder 20 that may implement the techniques of this disclosure. FIG. 5 is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder 20 in the context of HEVC coding. However, the techniques of this disclosure may be applicable to other coding standards or methods.

[0102] Video encoder 20 represents an example of a device that may be configured to perform techniques for palette-based video coding in accordance with various examples described in this disclosure. For example, video encoder 20 may be configured to selectively code various blocks of video data, such as CUs or PUs in HEVC coding, using either palette-based coding or non-palette based coding. Non-palette based coding modes may refer to various inter-predictive temporal coding modes or intra-predictive spatial coding modes, such as the various coding modes specified by HEVC Version 1. Video encoder 20, in one example, may be configured to generate a palette having entries indicating pixel values, select pixel values in a palette to represent pixels values of at least some positions of a block of video data, and signal information associating at least some of the positions of the block of video data with entries in the palette corresponding, respectively, to the selected pixel values. The signaled information may be used by video decoder 30 to decode video data.

[0103] In the example of FIG. 5, video encoder 20 includes a video data memory 101, a prediction processing unit 100, a residual generation unit 102, a transform processing unit 104, a quantization unit 106, an inverse quantization unit 108, an inverse transform processing unit 110, a reconstruction unit 112, a filter unit 114, a decoded picture buffer 116, and an entropy encoding unit 118. Prediction processing unit 100 includes an inter-prediction processing unit 120 and an intra-prediction processing unit 126. Inter-prediction processing unit 120 may include a motion estimation unit and a motion compensation unit (not shown). Video encoder 20 also includes a palette-based encoding unit 122 configured to perform various aspects of the palette-based coding techniques described in this disclosure. In other examples, video encoder 20 may include more, fewer, or different functional components.

[0104] Video data memory 101 may store video data to be encoded by the components of video encoder 20. The video data stored in video data memory 101 may be obtained, for example, from video source 18. Decoded picture buffer 116 may be a reference picture memory that stores reference video data for use in encoding video data by video encoder 20, e.g., in intra- or inter-coding modes. Video data memory 101 and decoded picture buffer 116 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. Video data memory 101 and decoded picture buffer 116 may be provided by the same memory device or separate memory devices. In various examples, video data memory 101 may be on-chip with other components of video encoder 20, or off-chip relative to those components.

[0105] Video encoder 20 may receive video data. Video encoder 20 may encode each CTU in a slice of a picture of the video data. Each of the CTUs may be associated with equally-sized luma coding tree blocks (CTBs) and corresponding CTBs of the picture. As part of encoding a CTU, prediction processing unit 100 may perform quad-tree partitioning to divide the CTBs of the CTU into progressively-smaller blocks. The smaller blocks may be coding blocks of CUs. For example, prediction processing unit 100 may partition a CTB associated with a CTU into four equally-sized sub-blocks, partition one or more of the sub-blocks into four equally-sized sub-sub-blocks, and so on.

[0106] Video encoder 20 may encode CUs of a CTU to generate encoded representations of the CUs (i.e., coded CUs). As part of encoding a CU, prediction processing unit 100 may partition the coding blocks associated with the CU among one or more PUs of the CU. Thus, each PU may be associated with a luma prediction block and corresponding chroma prediction blocks. Video encoder 20 and video decoder 30 may support PUs having various sizes. As indicated above, the size of a CU may refer to the size of the luma coding block of the CU and the size of a PU may refer to the size of a luma prediction block of the PU. Assuming that the size of a particular CU is 2Nx2N, video encoder 20 and video decoder 30 may support PU sizes of 2Nx2N or NxN for intra prediction, and symmetric PU sizes of 2Nx2N, 2NxN, Nx2N, NxN, or similar for inter prediction. Video encoder 20 and video decoder 30 may also support asymmetric partitioning for PU sizes of 2NxnU, 2NxnD, nLx2N, and nRx2N for inter prediction.

[0107] Inter-prediction processing unit 120 may generate predictive data for a PU by performing inter prediction on each PU of a CU. The predictive data for the PU may include predictive blocks of the PU and motion information for the PU. Inter-prediction processing unit 120 may perform different operations for a PU of a CU depending on whether the PU is in an I slice, a P slice, or a B slice. In an I slice, all PUs are intra predicted. Hence, if the PU is in an I slice, inter-prediction processing unit 120 does not perform inter prediction on the PU. Thus, for blocks encoded in I-mode, the predicted block is formed using spatial prediction from previously-encoded neighboring blocks within the same frame.

[0108] If a PU is in a P slice, the motion estimation unit of inter-prediction processing unit 120 may search the reference pictures in a list of reference pictures (e.g., "RefPicList0") for a reference region for the PU. The reference region for the PU may be a region, within a reference picture, containing sample blocks that most closely corresponds to the prediction blocks of the PU. Inter-prediction processing unit 120 may generate a reference index indicating a position in RefPicList0 of the reference picture containing the reference region for the PU. In addition, inter-prediction processing unit 120 may generate a motion vector indicating a spatial displacement between a prediction block of the PU and a reference location associated with the reference region. For instance, the motion vector may be a two-dimensional vector indicating an offset from the coordinates in the current decoded picture to coordinates in a reference picture. Inter-prediction processing unit 120 may output the reference index and the motion vector as the motion information of the PU. Inter-prediction processing unit 120 of inter-prediction processing unit 120 may generate the predictive blocks of the PU based on actual or interpolated samples at the reference location indicated by the motion vector of the PU.

[0109] If a PU is in a B slice, inter-prediction processing unit 120 may perform uni-prediction or bi-prediction for the PU. To perform uni-prediction for the PU, inter-prediction processing unit 120 may search the reference pictures of RefPicList0 or a second reference picture list ("RefPicList1") for a reference region for the PU. Inter-prediction processing unit 120 may output, as the motion information of the PU, a reference index indicating a position in RefPicList0 or RefPicList1 of the reference picture containing the reference region, a motion vector indicating a spatial displacement between a sample block of the PU and a reference location associated with the reference region, and one or more prediction direction indicators indicating whether

the reference picture is in RefPicList0 or RefPicList1. Inter-prediction processing unit 120 may generate the predictive blocks of the PU based at least in part on actual or interpolated samples at the reference region indicated by the motion vector of the PU.

[0110] To perform bi-directional inter prediction for a PU, inter-prediction processing unit 120 may search the reference pictures in RefPicList0 for a reference region for the PU and may also search the reference pictures in RefPicList1 for another reference region for the PU. Inter-prediction processing unit 120 may generate reference picture indexes indicating positions in RefPicList0 and RefPicList1 of the reference pictures containing the reference regions. In addition, inter-prediction processing unit 120 may generate motion vectors indicating spatial displacements between the reference locations associated with the reference regions and a prediction block of the PU. The motion information of the PU may include the reference indexes and the motion vectors of the PU. Inter-prediction processing unit 120 may generate the predictive blocks of the PU based at least in part on actual or interpolated samples at the reference region indicated by the motion vector of the PU.

[0111] In accordance with various examples of this disclosure, video encoder 20 may be configured to perform palette-based coding. With respect to the HEVC framework, as an example, the palette-based coding techniques may be configured to be used as a CU mode. In other examples, the palette-based coding techniques may be configured to be used as a PU mode in the framework of HEVC. Accordingly, the disclosed processes described herein (throughout this disclosure) in the context of a CU mode may, additionally or alternatively, apply to PUs. However, these HEVC-based examples should not be considered a restriction or limitation of the palette-based coding techniques described herein, as such techniques may be applied to work independently or as part of other existing or yet to be developed systems/standards. In these cases, the unit for palette coding can be square blocks, rectangular blocks or even regions of non-rectangular shape.

[0112] Palette-based encoding unit 122, for example, may perform palette-based encoding, e.g., for a CU or PU. For example, palette-based encoding unit 122 may be configured to generate a palette including entries, each indicating at least one sample value, select sample values in a palette to represent sample values of at least some positions of a block of video data, and signal information associating at least some of the positions of the block of video data with entries in the palette corresponding, respectively, to the selected sample values. Although various functions are described as

being performed by palette-based encoding unit 122, some or all of such functions may be performed by other processing units, or a combination of different processing units. In other examples, palette-based encoding unit 122 may be outside prediction processing unit 100.

[0113] In accordance with some examples of this disclosure, palette-based encoding unit 122 may determine a palette predictor list including or consisting of one or more candidates. Each respective candidate in the palette predictor list may specify at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Each of the one or more reconstructed neighboring pixels may be in a line above or a column left of a current block of the video data. Palette-based encoding unit 122 may include, in a palette for the current block, at least one candidate in the palette predictor list. Furthermore, palette-based encoding unit 122 may signal, in a bitstream, one or more syntax elements indicating an index value for a pixel location within the current block, the index value corresponding to an entry in the palette.

[0114] Intra-prediction processing unit 126 may generate predictive data for a PU by performing intra prediction on the PU. The predictive data for the PU may include predictive blocks for the PU and various syntax elements. Intra-prediction processing unit 126 may perform intra prediction on PUs in I slices, P slices, and B slices.

[0115] To perform intra prediction on a PU, intra-prediction processing unit 126 may use multiple intra prediction modes to generate multiple sets of predictive data for the PU. In some examples, intra-prediction processing unit 126 may determine samples of a predictive block based on values of reference samples in neighboring blocks. The neighboring blocks may be above, above and to the right, above and to the left, or to the left of the PU, assuming a left-to-right, top-to-bottom encoding order for PUs, CUs, and CTUs. Intra-prediction processing unit 126 may use various numbers of intra prediction modes, e.g., 33 directional intra prediction modes. In some examples, the number of intra prediction modes may depend on the size of the region associated with the PU.

[0116] Prediction processing unit 100 may select the predictive data for PUs of a CU from among the predictive data generated by inter-prediction processing unit 120 for the PUs or the predictive data generated by intra-prediction processing unit 126 for the PUs. In some examples, prediction processing unit 100 selects the predictive data for the PUs of the CU based on rate/distortion metrics of the sets of predictive data. The predictive

blocks of the selected predictive data may be referred to herein as the selected predictive blocks.

[0117] Residual generation unit 102 may generate, based on the coding blocks (e.g., luma, Cb and Cr coding blocks) of a CU and the selected predictive blocks (e.g., predictive luma, Cb and Cr blocks) of the PUs of the CU, residual blocks (e.g., luma, Cb and Cr residual blocks) of the CU. For instance, residual generation unit 102 may generate the residual blocks of the CU such that each sample in the residual blocks has a value equal to a difference between a sample in a coding block of the CU and a corresponding sample in a corresponding selected predictive sample block of a PU of the CU.

[0118] Transform processing unit 104 may perform quad-tree partitioning to partition the residual blocks associated with a CU into transform blocks associated with TUs of the CU. Thus, a TU may be associated with a luma transform block and two chroma transform blocks. The sizes and positions of the luma and chroma transform blocks of TUs of a CU may or may not be based on the sizes and positions of prediction blocks of the PUs of the CU. A quad-tree structure known as a "residual quad-tree" (RQT) may include nodes associated with each of the regions. The TUs of a CU may correspond to leaf nodes of the RQT.

[0119] Transform processing unit 104 may generate transform coefficient blocks for each TU of a CU by applying one or more transforms to the transform blocks of the TU. Transform processing unit 104 may apply various transforms to a transform block of a TU. For example, transform processing unit 104 may apply a discrete cosine transform (DCT), a directional transform, or a conceptually similar transform to a transform block. In some examples, transform processing unit 104 does not apply transforms to a transform block. In such examples, the transform block may be treated as a transform coefficient block.

[0120] Quantization unit 106 may quantize the transform coefficients in a coefficient block. The quantization process may reduce the bit depth associated with some or all of the transform coefficients. 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. Quantization unit 106 may quantize a coefficient block associated with a TU of a CU based on a quantization parameter (QP) value associated with the CU. Video encoder 20 may adjust the degree of quantization applied to the coefficient blocks associated with a CU by adjusting the QP value associated with the CU. Quantization may introduce loss

of information, thus quantized transform coefficients may have lower precision than the original ones.

[0121] Inverse quantization unit 108 and inverse transform processing unit 110 may apply inverse quantization and inverse transforms to a coefficient block, respectively, to reconstruct a residual block from the coefficient block. Reconstruction unit 112 may add the reconstructed residual block to corresponding samples from one or more predictive sample blocks generated by prediction processing unit 100 to produce a reconstructed transform block associated with a TU. By reconstructing transform blocks for each TU of a CU in this way, video encoder 20 may reconstruct the coding blocks of the CU.

[0122] Filter unit 114 may perform one or more deblocking operations to reduce blocking artifacts in the coding blocks associated with a CU. Decoded picture buffer 116 may store the reconstructed coding blocks after filter unit 114 performs the one or more deblocking operations on the reconstructed coding blocks. Inter-prediction processing unit 120 may use a reference picture containing the reconstructed coding blocks to perform inter prediction on PUs of other pictures. In addition, intra-prediction processing unit 126 may use reconstructed coding blocks in decoded picture buffer 116 to perform intra prediction on other PUs in the same picture as the CU.

[0123] Entropy encoding unit 118 may receive data from other functional components of video encoder 20. For example, entropy encoding unit 118 may receive coefficient blocks from quantization unit 106 and may receive syntax elements from prediction processing unit 100. Entropy encoding unit 118 may perform one or more entropy encoding operations on the data to generate entropy-encoded data. For example, entropy encoding unit 118 may perform a CABAC operation, a context-adaptive variable length coding (CAVLC) operation, a variable-to-variable (V2V) length coding operation, a syntax-based context-adaptive binary arithmetic coding (SBAC) operation, a Probability Interval Partitioning Entropy (PIPE) coding operation, an Exponential-Golomb encoding operation, or another type of entropy encoding operation on the data. Video encoder 20 may output a bitstream that includes entropy-encoded data generated by entropy encoding unit 118. For instance, the bitstream may include data that represents a RQT for a CU.

[0124] FIG. 6 is a block diagram illustrating an example video decoder 30 that is configured to implement the techniques of this disclosure. FIG. 6 is provided for purposes of explanation and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes

video decoder 30 in the context of HEVC coding. However, the techniques of this disclosure may be applicable to other coding standards or methods.

[0125] Video decoder 30 represents an example of a device configured to perform techniques for palette-based video coding in accordance with various examples described in this disclosure. For example, video decoder 30 may be configured to selectively decode various blocks of video data, such as CUs or PUs in HEVC coding, using either palette-based coding or non-palette based coding. Non-palette based coding modes may refer to various inter-predictive temporal coding modes or intra-predictive spatial coding modes, such as the various coding modes specified by HEVC Version 1. Video decoder 30, in one example, may be configured to generate a palette having entries indicating pixel values, receive information associating at least some positions of a block of video data with entries in the palette, select pixel values in the palette based on the information, and reconstruct pixel values of the block based on the selected pixel values.

[0126] In the example of FIG. 6, video decoder 30 includes an entropy decoding unit 150, video data memory 151, a prediction processing unit 152, an inverse quantization unit 154, an inverse transform processing unit 156, a reconstruction unit 158, a filter unit 160, and a decoded picture buffer 162. Prediction processing unit 152 includes a motion compensation unit 164 and an intra-prediction processing unit 166. Video decoder 30 also includes a palette-based decoding unit 165 configured to perform various aspects of the palette-based coding techniques described in this disclosure. In other examples, video decoder 30 may include more, fewer, or different functional components.

[0127] Video data memory 151 may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 30. The video data stored in video data memory 151 may be obtained, for example, from channel 16, e.g., from a local video source, such as a camera, via wired or wireless network communication of video data, or by accessing physical data storage media. Video data memory 151 may form a coded picture buffer (CPB) that stores encoded video data from an encoded video bitstream. Decoded picture buffer 162 may be a reference picture memory that stores reference video data for use in decoding video data by video decoder 30, e.g., in intra- or inter-coding modes. Video data memory 151 and decoded picture buffer 162 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. Video data memory 151 and decoded picture buffer 162 may be provided by the same memory device or separate memory devices. In various examples, video data memory 151 may be on-chip with other components of video decoder 30, or off-chip relative to those components.

[0128] A coded picture buffer (CPB) may receive and store encoded video data (e.g., NAL units) of a bitstream. Entropy decoding unit 150 may receive encoded video data (e.g., NAL units) from the CPB and may parse the NAL units to decode syntax elements. Entropy decoding unit 150 may entropy decode entropy-encoded syntax elements in the NAL units. Prediction processing unit 152, inverse quantization unit 154, inverse transform processing unit 156, reconstruction unit 158, and filter unit 160 may generate decoded video data based on the syntax elements extracted from the bitstream.

[0129] The NAL units of the bitstream may include coded slice NAL units. As part of decoding the bitstream, entropy decoding unit 150 may extract and entropy decode syntax elements from the coded slice NAL units. Each of the coded slices may include a slice header and slice data. The slice header may contain syntax elements pertaining to a slice. The syntax elements in the slice header may include a syntax element that identifies a PPS associated with a picture that contains the slice.

[0130] In addition to decoding syntax elements from the bitstream, video decoder 30 may perform a reconstruction operation on a non-partitioned CU. To perform the reconstruction operation on a non-partitioned CU, video decoder 30 may perform a reconstruction operation on each TU of the CU. By performing the reconstruction operation for each TU of the CU, video decoder 30 may reconstruct residual blocks of the CU.

[0131] As part of performing a reconstruction operation on a TU of a CU, inverse quantization unit 154 may inverse quantize, i.e., de-quantize, coefficient blocks associated with the TU. Inverse quantization unit 154 may use a QP value associated with the CU of the TU to determine a degree of quantization and, likewise, a degree of inverse quantization for inverse quantization unit 154 to apply. That is, the compression ratio, i.e., the ratio of the number of bits used to represent original sequence and the compressed one, may be controlled by adjusting the value of the QP used when quantizing transform coefficients. The compression ratio may also depend on the method of entropy coding employed.

[0132] After inverse quantization unit 154 inverse quantizes a coefficient block, inverse transform processing unit 156 may apply one or more inverse transforms to the coefficient block in order to generate a residual block associated with the TU. For example, inverse transform processing unit 156 may apply an inverse DCT, an inverse integer transform, an inverse Karhunen-Loeve transform (KLT), an inverse rotational transform, an inverse directional transform, or another inverse transform to the coefficient block.

[0133] If a PU is encoded using intra prediction, intra-prediction processing unit 166 may perform intra prediction to generate predictive blocks for the PU. Intra-prediction processing unit 166 may use an intra prediction mode to generate the predictive blocks (e.g., predictive luma, Cb and Cr blocks) for the PU based on the prediction blocks of spatially-neighboring PUs. Intra-prediction processing unit 166 may determine the intra prediction mode for the PU based on one or more syntax elements decoded from the bitstream.

[0134] Prediction processing unit 152 may construct a first reference picture list (RefPicList0) and a second reference picture list (RefPicList1) based on syntax elements extracted from the bitstream. Furthermore, if a PU is encoded using inter prediction, entropy decoding unit 150 may extract motion information for the PU. Motion compensation unit 164 may determine, based on the motion information of the PU, one or more reference regions for the PU. Motion compensation unit 164 may generate, based on samples blocks at the one or more reference blocks for the PU, predictive blocks (e.g., predictive luma, Cb and Cr blocks) for the PU.

[0135] Reconstruction unit 158 may use the transform blocks (e.g., luma, Cb and Cr transform blocks) of TUs of a CU and the predictive blocks (e.g., predictive luma, Cb and Cr blocks) of the PUs of the CU, i.e., either intra-prediction data or inter-prediction data, as applicable, to reconstruct the coding blocks (e.g., luma, Cb and Cr coding blocks) of the CU. For example, reconstruction unit 158 may add samples of the transform blocks (e.g., luma, Cb and Cr transform blocks) to corresponding samples of the predictive blocks (e.g., predictive luma, Cb and Cr blocks) to reconstruct the coding blocks (e.g., luma, Cb and Cr coding blocks) of the CU.

[0136] Filter unit 160 may perform a deblocking operation to reduce blocking artifacts associated with the coding blocks (e.g., luma, Cb and Cr coding blocks) of the CU. Video decoder 30 may store the coding blocks (e.g., luma, Cb and Cr coding blocks) of the CU in decoded picture buffer 162. Decoded picture buffer 162 may provide

reference pictures for subsequent motion compensation, intra prediction, and presentation on a display device, such as display device 32 of FIG. 1. For instance, video decoder 30 may perform, based on the blocks (e.g., luma, Cb and Cr blocks) in decoded picture buffer 162, intra prediction or inter prediction operations on PUs of other CUs. In this way, video decoder 30 may extract, from the bitstream, transform coefficient levels of a significant coefficient block, inverse quantize the transform coefficient levels, apply a transform to the transform coefficient levels to generate a transform block, generate, based at least in part on the transform block, a coding block, and output the coding block for display.

[0137] Palette-based decoding unit 165, for example, may perform palette-based decoding when a palette-based decoding mode is selected, e.g., for a CU or PU. For example, palette-based decoding unit 165 may be configure to generate a palette having entries indicating pixel values, receive information associating at least some positions of a block of video data with entries in the palette, select pixel values in the palette based on the information, and reconstruct pixel values of the block based on the selected pixel values. Although various functions are described as being performed by palette-based decoding unit 165, some or all of such functions may be performed by other processing units, or a combination of different processing units.

[0138] In accordance with some examples of this disclosure, palette-based decoding unit 165 may determine a palette predictor list including or consisting of one or more candidates. Each respective candidate in the palette predictor list may specify at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Each of the one or more reconstructed neighboring pixels may be in a line above or a column left of a current block of the video data. Furthermore, palette-based decoding unit 165 may include, in a palette for the current block, at least one candidate in the palette predictor list. Palette-based decoding unit 165 may obtain, based on one or more syntax elements signaled in a bitstream, an index value for a pixel of the current block. Furthermore, palette-based decoding unit 165 may identify a palette entry in the palette that corresponds to the index value for the pixel. Additionally, palette-based decoding unit 165 may determine, based on the at least one sample value specified by the identified palette entry, at least one reconstructed sample value of the pixel. By determining reconstructed sample values of pixels of the current block, palette-based decoding unit 165 may reconstruct the current block.

[0139] FIG. 7 is a flowchart illustrating an example decoding operation, in accordance with a technique of this disclosure. The flowcharts of this disclosure are provided as examples. In other examples, the operations shown in the flowcharts may include more, fewer, or different actions. Moreover, in other examples, actions may be performed in different orders or in parallel. FIG. 7 is described from the context of video decoder 30, although other types of devices or decoders may also be used to perform the actions shown in FIG. 7.

[0140] In the example of FIG. 7, video decoder 30 determines a palette predictor list comprising or consisting of one or more candidates (700). Each respective candidate in the palette predictor list may specify at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data. Video decoder 30 may include, in a palette for the current block, at least one candidate in the palette predictor list (702).

[0141] Furthermore, video decoder 30 may obtain, based on one or more syntax elements (e.g., a palette_index syntax element) signaled in a bitstream, an index value for a pixel of the current block (704). Additionally, video decoder 30 may identify a palette entry in the palette that corresponds to the index value for the pixel (706). Video decoder 30 may determine, based on the at least one sample value specified by the identified palette entry, at least one reconstructed sample value of the pixel (708).

[0142] FIG. 8 is a flowchart illustrating an example encoding operation, in accordance with a technique of this disclosure. FIG. 7 is described from the context of video encoder 20, although other types of devices or encoders may also be used to perform the actions shown in FIG. 8. In the example of FIG. 8, video encoder 20 determines a palette predictor list comprising or consisting one or more candidates (800). In the example of FIG. 8, each respective candidate in the palette predictor list specifies at least one value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels. Furthermore, in the example of FIG. 8, each of the one or more reconstructed neighboring pixels is adjacent to a current block of the video data.

[0143] In the example of FIG. 8, video encoder 20 includes, in a palette for the current block, at least one candidate in the palette predictor list (802). Video encoder 20 signals, in a bitstream, one or more syntax elements indicating an index value for a pixel of the current block, the index value corresponding to an entry in the palette (804).

[0144] FIG. 9 is a flowchart illustrating an example operation for generating a palette, in accordance with a technique of this disclosure. In the example of FIG. 9, a video coder (e.g., a video encoder or a video decoder) generates an original palette predictor list (900). FIG. 10, described in detail elsewhere in this disclosure, is an example operation a video decoder may use to generate the original palette predictor list. FIG.12, described in detail elsewhere in this disclosure, is an example operation a video encoder may use to generate the original palette predictor list.

[0145] Furthermore, in the example of FIG. 9, the video coder generates an extra palette predictor list (902). FIG. 13, described in detail elsewhere in this disclosure, is an example operation of a video encoder to generate the extra palette predictor list. FIG. 11, described in detail elsewhere in this disclosure, is an example operation of a video decoder to generate the extra palette predictor list.

[0146] In the example of FIG. 9, the video coder applies a pruning process to the original palette predictor list and the extra palette predictor list (904). In other examples, the video coder does not perform the pruning process. The pruning process may remove, from the extra palette predictor list, a particular candidate in the extra palette predictor list if the particular candidate matches a candidate in the original palette predictor list. Furthermore, in some examples, as part of applying the pruning process, the video coder determines, based on a metric of similarity of a first candidate in the extra palette predictor list and a second candidate in the original palette predictor list, whether to remove the first candidate from the extra palette predictor list. The video coder may remove the first candidate from the extra palette predictor list responsive to the determination.

[0147] Furthermore, in the example of FIG. 9, the video coder obtains, from the bitstream, syntax elements (e.g., palette_entries syntax elements) specifying one or more additional palette entries (906). In the example of FIG. 9, the video coder includes, in the palette for the current block, palette entries corresponding to remaining candidates in the original palette predictor list, the extra palette predictor list, and the additional palette entries (908). In some examples, the video coder does not obtain any syntax elements specifying additional palette entries or include such additional palette entries in the palette.

[0148] In some examples, the video coder orders the palette entries in the palette for the current block as follows: palette entries specifying candidates in the extra palette predictor list, palette entries specifying candidates in the original palette predictor list,

and the additional palette entries. In other examples, the video coder orders the palette entries in the palette for the current block as follows: palette entries specifying candidates in the original palette predictor list, palette entries specifying candidates in the extra palette predictor list, and the additional palette entries.

[0149] FIG. 10 is a flowchart illustrating an example operation of video decoder 30 to generate an original palette predictor list, in accordance with a technique of this disclosure. FIG. 10 is described from the context of video decoder 30, although other types of devices or decoders may also be used to perform the actions shown in FIG. 10. The operation of FIG. 10 may be an instance of action 900 of FIG. 9 to determine an original palette predictor list. In the example of FIG. 10, for each respective palette entry of a previously-coded block, video decoder 30 includes a respective candidate in the original palette predictor list (1000). The respective candidate specifies the same pixel as the respective palette entry. Thus, video decoder 30 may determine the original palette predictor list such that the original palette predictor list comprises one or more candidates, each respective candidate in the original palette predictor list specifying a palette entry in a palette for the previously-coded block of the video data. In some examples, the original palette predictor list may also include one or more candidates based on entries in a palette for a block coded prior to the previously-coded block, as described in JCTVC-R0082.

[0150] Furthermore, video decoder 30 may obtain, from the bitstream, a binary vector for the original palette predictor list (1002). In the example of FIG. 10, the binary vector for the original palette predictor list comprises one or more binary vector syntax elements (e.g., previous_palette_entry_flag syntax elements). Each of the one or more binary vector syntax elements corresponds to a different candidate in the original palette predictor list.

[0151] In the example of FIG. 10, video decoder 30 may perform actions (1004)-(1008) for each respective candidate in the original palette predictor list. Particularly, video decoder 30 may determine whether the binary vector syntax element corresponding to the respective candidate in the original palette predictor list indicates the respective candidate is reused in the current block (1004). Responsive to determining the binary vector syntax element corresponding to the respective candidate in the original palette predictor list indicates the respective candidate is reused in the current block ("YES" of 1004), video decoder 30 retains, in the original palette predictor list, the respective candidate (1006). Otherwise, responsive to determining the binary vector syntax

element corresponding to the respective candidate in the original palette predictor list indicates the respective candidate is not reused in the current block ("NO" of 1004), video decoder 30 omits, from the original palette predictor list, the respective candidate (1008). For example, the binary vector "01001..." indicates that the second and fifth candidates in the original palette predictor list are retained in the original palette predictor list and the first, third, and fourth candidates in the original palette predictor list are ultimately omitted from the original palette predictor list.

[0152] Because video decoder 30 omits the respective candidate from the original palette predictor list, video decoder 30 effectively determines that the respective candidate is not included in the palette of the current block. Thus, in the example of FIG. 10, for each respective candidate in the original palette predictor list, video decoder 30 may determine, based on the binary vector syntax element corresponding to the respective candidate in the original palette predictor list, the respective candidate in the original palette predictor list is not included in the palette for the current block.

[0153] FIG. 11 is a flowchart illustrating an example operation of video decoder 30 to generate an extra palette predictor list, in accordance with a technique of this disclosure. Although FIG. 11 is described from the context of video decoder 30, other types of devices or decoders may also be used to perform the actions shown in FIG. 11. The operation of FIG. 11 may be an instance of action 902 of FIG. 9 to determine an extra palette predictor list. In the example of FIG. 11, for each respective neighbor pixel of a predefined set of neighbor pixels in a line above or a column left of a current block, video decoder 30 generates, in the extra palette predictor list, a respective candidate specifying the value of the respective neighbor pixel (1100).

[0154] Furthermore, video decoder 30 may remove redundant candidates from the extra palette predictor list (1102). For instance, video decoder 30 may remove matching candidates from the extra palette predictor list. In one example, as part of removing redundant candidates from the extra palette predictor list, video decoder 30 may determine, based on a metric of similarity of a first candidate in the palette predictor list and a second candidate in the palette predictor list, whether to remove the first candidate from the palette predictor list. In this example, video decoder 30 may remove the first candidate from the palette predictor list responsive to the determination.

[0155] In the example of FIG. 11, video decoder 30 may determine whether a number of candidates in the extra palette predictor list exceeds a threshold (1104). Responsive to determining the number of candidates in the extra palette predictor list exceeds the

threshold ("YES" of 1104), video decoder 30 may truncate the extra palette predictor list such that a size of the extra palette predictor list is limited to a particular number of candidates (e.g., the threshold) (1106). Responsive to determining the number of candidates in the extra palette predictor list does not exceed the threshold ("NO" of 1104), video decoder 30 may refrain from truncating the extra palette predictor list (1108).

[0156] Additionally, video decoder 30 may obtain, from a bitstream, a binary vector for the extra palette predictor list (1110). The binary vector for the extra palette predictor list includes one or more syntax elements. Each of the one or more syntax elements corresponds to a different candidate in the palette predictor list.

[0157] In the example operation of FIG. 11, video decoder 30 may perform actions (1112) through (1116) for each respective candidate in the extra palette predictor list. Particularly, video decoder 30 determines whether the syntax element corresponding to the respective candidate indicates the respective candidate is reused in the current block (1112). Responsive to determining the syntax element corresponding to the respective candidate indicates the respective candidate is reused in the current block ("YES" of 1112), video decoder 30 retains, in the extra palette predictor list, the respective candidate (1114). Responsive to determining the syntax element corresponding to the respective candidate indicates the candidate is not reused in the current block ("NO" of 1112), video decoder 30 may omit, from the extra palette predictor list, the respective candidate (1116). For example, the binary vector "01001…" indicates the second and fifth candidates in the extra palette predictor list (e.g., the second and fifth neighboring reconstructed pixels) are retained in the extra palette predictor list and the first, third, and fourth candidates in the extra palette predictor list (e.g., the first, third, and fourth neighboring reconstructed pixels) are ultimately omitted from the extra palette predictor list.

[0158] Because video decoder 30 omits the respective candidate from the extra palette predictor list, video decoder 30 effectively determines the respective candidate is not included in the palette of the current block. Thus, in the example of FIG. 11, for each respective candidate in the extra palette predictor list, video decoder 30 may determine, based on the binary vector syntax element corresponding to the respective candidate in the extra palette predictor list, the respective candidate in the extra palette predictor list is not included in the palette for the current block.

[0159] FIG. 12 is a flowchart illustrating an example operation of video encoder 20 to generate an original palette predictor list, in accordance with a technique of this disclosure. Although FIG. 12 is described from the context of video encoder 20, other types of devices or decoders may also be used to perform the actions shown in FIG. 12. The operation of FIG. 12 may be an instance of action 900 of FIG. 9 to determine an original palette predictor list. In the example of FIG. 12, for each respective palette entry of a previously-coded block, video encoder 20 includes a respective candidate in the original palette predictor list (1200). The respective candidate specifies the same pixel as the respective palette entry. Thus, video encoder 20 may determine the original palette predictor list such that the original palette predictor list comprises one or more candidates, each respective candidate in the original palette predictor list specifying a palette entry in the palette for the previously-coded block of the video data. In some examples, the original palette predictor list may also include one or more candidates based on entries in a palette for a block coded prior to the previously-coded block, as described in JCTVC-R0082.

[0160] Furthermore, the video encoder 20 may signal, in the bitstream, a binary vector for the original palette predictor list (1202). In the example of FIG. 12, the binary vector for the original palette predictor list comprises one or more binary vector syntax elements. Each of the one or more binary vector syntax elements corresponds to a different candidate in the original palette predictor list.

[0161] In the example of FIG. 12, video encoder 20 may perform actions (1204)-(1208) for each respective candidate in the original palette predictor list. Particularly, video encoder 20 may determine whether the binary vector syntax element corresponding to the respective candidate in the original palette predictor list indicates respective candidate is reused in the current block (1204). Responsive to determining the binary vector syntax element corresponding to the respective candidate in the original palette predictor list indicates that the candidate is reused in the current block ("YES" of 1204), video encoder 20 retains, in the original palette predictor list, the respective candidate (1206).

[0162] Otherwise, responsive to determining the binary vector syntax element corresponding to the respective candidate in the original palette predictor list indicates the candidate is not reused in the current block ("NO" of 1204), video encoder 20 omits, from the original palette predictor list, the respective candidate (1208). Because video encoder 20 omits the respective candidate from the original palette predictor list, video

encoder 20 effectively determines the respective candidate is not included in the palette of the current block. Thus, in the example of FIG. 12, for each respective candidate in the original palette predictor list, video encoder 20 may determine, based on the binary vector syntax element corresponding to the respective candidate in the original palette predictor list, the respective candidate in the original palette predictor list is not included in the palette for the current block.

[0163] FIG. 13 is a flowchart illustrating an example operation of video encoder 20 to generate an extra palette predictor list, in accordance with a technique of this disclosure. Although FIG. 13 is described from the context of video encoder 20, other types of devices or decoders may also be used to perform the actions shown in FIG. 13. The operation of FIG. 13 may be an instance of action 902 of FIG. 9 to determine an extra palette predictor list. In the example of FIG. 13, for each respective neighbor pixel of a predefined set of neighbor pixels in a line above or a column left of a current block, video encoder 20 generates, in the extra palette predictor list, a respective candidate specifying the value of the respective neighbor pixel (1300).

[0164] Furthermore, video encoder 20 may remove redundant candidates from the extra palette predictor list (1302). For instance, video encoder 20 may remove matching candidates from the extra palette predictor list. In one example, as part of removing redundant candidates from the extra palette predictor list, video encoder 20 may determine, based on a metric of similarity of a first candidate in the palette predictor list and a second candidate in the palette predictor list, whether to remove the first candidate from the palette predictor list. In this example, video encoder 20 may remove the first candidate from the palette predictor list responsive to the determination.

[0165] In the example of FIG. 13, video encoder 20 may determine whether a number of candidates in the extra palette predictor list exceeds a threshold (1304). Responsive to determining the number of candidates in the extra palette predictor list exceeds the threshold ("YES" of 1304), video encoder 20 may truncate the extra palette predictor list such that a size of the extra palette predictor list is limited to a particular number of candidates (e.g., the threshold) (1306). Responsive to determining the number of candidates in the extra palette predictor list does not exceed the threshold ("NO" of 1304), video encoder 20 may refrain from truncating the extra palette predictor list (1308).

[0166] Additionally, video encoder 20 may signal, in a bitstream, a binary vector for the extra palette predictor list (1310). The binary vector for the extra palette predictor list

includes one or more syntax elements. Each of the one or more syntax elements corresponds to a different candidate in the palette predictor list.

[0167] In the example operation of FIG. 13, video encoder 20 may perform actions (1312) through (1316) for each respective candidate in the extra palette predictor list. Particularly, video encoder 20 determines whether the syntax element corresponding to the respective candidate indicates the respective candidate is re-used in the current block (1312). Responsive to determining the syntax element corresponding to the respective candidate indicates the respective candidate is reused in the current block ("YES" of 1312), video encoder 20 retains, in the extra palette predictor list, the respective candidate (1314).

[0168] Responsive to determining the syntax element corresponding to the respective candidate indicates the candidate is not reused in the current block ("NO" of 1312), video encoder 20 may omit, from the extra palette predictor list, the respective candidate (1316). Because video encoder 20 omits the respective candidate from the extra palette predictor list, video encoder 20 effectively determines the respective candidate is not included in the palette of the current block. Thus, in the example of FIG. 13, for each respective candidate in the extra palette predictor list, video encoder 20 may determine, based on the binary vector syntax element corresponding to the respective candidate in the extra palette predictor list, the respective candidate in the extra palette predictor list is not included in the palette for the current block.

[0169] The following paragraphs list several examples of this disclosure.

[0170] Example 1. A method of decoding video data, the method comprising: determining a palette predictor list that comprises one or more candidates, each respective candidate in the palette predictor list specifying a value of a different reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is adjacent to a current block of the video data; obtaining, from a bitstream, a binary vector comprising one or more syntax elements, wherein each of the one or more syntax elements corresponds to a different candidate in the palette predictor list; for each respective candidate in the palette predictor list: if the syntax element corresponding to the respective candidate indicates that a palette for a current block of the video data includes the value specified by the respective candidate, including, in the palette, a palette entry specifying the value specified by the respective candidate; and if the syntax element corresponding to the respective candidate indicates that the palette does not

include the value specified by the respective candidate, omitting, from the palette, the palette entry specifying the value specified by the respective candidate; obtaining, based on one or more additional syntax elements signaled in a bitstream, an index value for a sample of the current block; identifying a palette entry in the palette that corresponds to the index value for the sample; and determining, based on a pixel value specified by the identified palette entry, a reconstructed value of the sample.

[0171] Example 2. The method of example 1, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from a line above the current block.

[0172] Example 3. The method of examples 1 or 2, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from a column left of the current block.

[0173] Example 4. The method of example 1, wherein the reconstructed neighboring pixels include at least one of: a below-left reconstructed neighboring pixel, a left reconstructed neighboring pixel, and above-left reconstructed neighboring pixel, an above reconstructed neighboring pixel, or an above-right reconstructed neighboring pixel.

[0174] Example 5. The method of example 1, wherein the reconstructed neighboring pixels include each of: a below-left reconstructed neighboring pixel, a left reconstructed neighboring pixel, and above-left reconstructed neighboring pixel, an above reconstructed neighboring pixel, and an above-right reconstructed neighboring pixel.

[0175] Example 6. The method of any of examples 1-5, wherein generating the first palette predictor list comprises: removing one or more redundant candidates from the first palette predictor list.

[0176] Example 7. The method of any of examples 1-6, wherein generating the first palette predictor list comprises: determining, based on a metric of similarity of a first candidate in the palette predictor list and a second candidate in the palette predictor list, whether to remove the first candidate from the palette predictor list; and removing the first candidate from the palette predictor list responsive to the determination.

[0177] Example 8. The method of any of examples 1-7, wherein the one or more neighboring pixels include a plurality of neighboring pixels, and generating the first palette predictor list comprises: ranking two or more of the plurality of reconstructed neighboring pixels according to frequency of values of the plurality of the reconstructed neighboring pixels.

[0178] Example 9. The method of any of examples 1-8, wherein determining the palette predictor list comprises: truncating the palette predictor list such that a size of the palette predictor list is limited to a particular number of candidates.

[0179] Example 10. The method of any of examples 1-9, wherein in-loop filtering is not applied to the values of the one or more reconstructed neighboring pixels specified by the candidates in the palette predictor list.

[0180] Example 11. The method of any of examples 1-10, wherein the palette predictor list is a first palette predictor list and the binary vector is a first binary vector, the method further comprising: determining a second palette predictor list that comprises one or more candidates, each respective candidate in the second palette predictor list specifying a palette entry in a palette for a previously-coded block of the video data; obtaining a second binary vector from the bitstream, the second binary vector comprising one or more second binary vector syntax elements, wherein each of the one or more second binary vector syntax elements corresponds to a different candidate in the second palette predictor list; and for each respective candidate in the second palette predictor list: if the second binary vector syntax element corresponding to the respective candidate in the second palette predictor list indicates that the palette for the current block includes the palette entry specified by the respective candidate, including, in the palette for the current block, the palette entry specified by the respective candidate; and if the second binary vector syntax element corresponding to the respective candidate in the second palette predictor list indicates that the palette for the current block does not include the palette entry specified by the respective candidate, omitting, from the palette for the current block, the palette entry specified by the respective candidate.

[0181] Example 12. The method of example 11, further comprising: applying a pruning process to the first and second palette predictor lists.

[0182] Example 13. The method of example 12, wherein applying the pruning process to the first and second palette predictor lists comprises: removing, from the first palette predictor list, a particular candidate in the first palette predictor list if the particular candidate matches a candidate in the second palette predictor list.

[0183] Example 14. The method of examples 12 or 13, wherein applying the pruning process to the first and second palette predictor lists comprises: determining, based on a metric of similarity of a first candidate in the first palette predictor list and a second candidate in the second palette predictor list, whether to remove the first candidate from

the first palette predictor list; and removing the first candidate from the first palette predictor list responsive to the determination.

[0184] Example 15. The method of any of examples 11-14, further comprising: obtaining, from the bitstream, syntax elements specifying one or more additional palette entries; and including, in the palette for the current block, the one or more additional palette entries.

[0185] Example 16. The method of example 15, wherein the palette entries in the palette for the current block are ordered as follows: palette entries that specify values specified by candidates in the first palette predictor list, the additional palette entries, and palette entries specified by the second palette predictor list.

[0186] Example 17. The method of example 15, wherein the palette entries in the palette for the current block are ordered as follows: palette entries that specify values specified by candidates in the first palette predictor list, palette entries specified by the second palette predictor list, and the additional palette entries.

[0187] Example 18. The method of any of examples 1-17, wherein candidates of the palette predictor list are not included in a palette predictor list used to determine a palette for a block of the video data processed after the current block.

[0188] Example 19. A method of encoding video data, the method comprising: determining a palette predictor list that comprises one or more candidates, each respective candidate in the palette predictor list specifying a value of a different reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is adjacent to a current block of the video data; signaling, in a bitstream, a binary vector comprising one or more syntax elements, wherein each of the one or more syntax elements corresponds to a different candidate in the palette predictor list; for each respective candidate in the palette predictor list: if the syntax element corresponding to the respective candidate indicates that a palette for a current block of the video data includes the value specified by the respective candidate, the palette includes a palette entry specifying the value specified by the respective candidate; and if the syntax element corresponding to the respective candidate indicates that the palette does not include the value specified by the respective candidate, the palette omits the palette entry specifying the value specified by the respective candidate; and signaling, in the bitstream, one or more syntax elements indicating an index value for a sample of the current block, the index value corresponding to an entry in the palette.

[0189] Example 20. The method of example 19, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from a line above the current block.

[0190] Example 21. The method of examples 19 or 20, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from a column left of the current block.

[0191] Example 22. The method of example 19, wherein the reconstructed neighboring pixels include at least one of: a below-left reconstructed neighboring pixel, a left reconstructed neighboring pixel, and above-left reconstructed neighboring pixel, an above reconstructed neighboring pixel, or an above-right reconstructed neighboring pixel.

[0192] Example 23. The method of example 19, wherein the reconstructed neighboring pixels include each of: a below-left reconstructed neighboring pixel, a left reconstructed neighboring pixel, and above-left reconstructed neighboring pixel, an above reconstructed neighboring pixel, and an above-right reconstructed neighboring pixel.

[0193] Example 24. The method of any of examples 19-23, wherein generating the first palette predictor list comprises: removing one or more redundant candidates from the first palette predictor list.

[0194] Example 25. The method of any of examples 19-24, wherein generating the first palette predictor list comprises: determining, based on a metric of similarity of a first candidate in the palette predictor list and a second candidate in the palette predictor list, whether to remove the first candidate from the palette predictor list; and removing the first candidate from the palette predictor list responsive to the determination.

[0195] Example 26. The method of any of examples 19-25, wherein the one or more neighboring pixels include a plurality of neighboring pixels, and generating the first palette predictor list comprises: ranking two or more of the plurality of reconstructed neighboring pixels according to frequency of values of the plurality of the reconstructed neighboring pixels.

[0196] Example 27. The method of any of examples 19-26, wherein determining the palette predictor list comprises: truncating the palette predictor list such that a size of the palette predictor list is limited to a particular number of candidates.

[0197] Example 28. The method of any of examples 19-27, wherein in-loop filtering is not applied to the values of the one or more reconstructed neighboring pixels specified by the candidates in the palette predictor list.

[0198] Example 29. The method of any of examples 19-28, wherein the palette predictor list is a first palette predictor list and the binary vector is a first binary vector, the method further comprising: determining a second palette predictor list that comprises one or more candidates, each respective candidate in the second palette predictor list specifying a palette entry in a palette for a previously-coded block of the video data; signaling a second binary vector in the bitstream, the second binary vector comprising one or more second binary vector syntax elements, wherein each of the one or more second binary vector syntax elements corresponds to a different candidate in the second palette predictor list; and for each respective candidate in the second palette predictor list: if the second binary vector syntax element corresponding to the respective candidate in the second palette predictor list indicates that the palette for the current block includes the palette entry specified by the respective candidate, the palette for the current block includes the palette entry specified by the respective candidate; and if the second binary vector syntax element corresponding to the respective candidate in the second palette predictor list indicates that the palette for the current block does not include the palette entry specified by the respective candidate, the palette for the current block omits the palette entry specified by the respective candidate.

[0199] Example 30. The method of example 29, further comprising: applying a pruning process to the first and second palette predictor lists.

[0200] Example 31. The method of example 30, wherein applying the pruning process to the first and second palette predictor lists comprises: removing, from the first palette predictor list, a particular candidate in the first palette predictor list if the particular candidate matches a candidate in the second palette predictor list.

[0201] Example 32. The method of examples 30 or 31, wherein applying the pruning process to the first and second palette predictor lists comprises: determining, based on a metric of similarity of a first candidate in the first palette predictor list and a second candidate in the second palette predictor list, whether to remove the first candidate from the first palette predictor list; and removing the first candidate from the first palette predictor list responsive to the determination.

[0202] Example 33. The method of any of examples 29-32, further comprising: including, in the palette for the current block, the one or more additional palette entries;

and signaling, in the bitstream, syntax elements specifying one or more additional palette entries.

[0203] Example 34. The method of example 33, wherein the palette entries in the palette for the current block are ordered as follows: palette entries that specify values specified by candidates in the first palette predictor list, the additional palette entries, and palette entries specified by the second palette predictor list.

[0204] Example 35. The method of example 33, wherein the palette entries in the palette for the current block are ordered as follows: palette entries that specify values specified by candidates in the first palette predictor list, palette entries specified by the second palette predictor list, and the additional palette entries.

[0205] Example 36. The method of any of examples 19-35, wherein candidates of the palette predictor list are not included in a palette predictor list used to determine a palette for a block of the video data processed after the current block.

[0206] Example 37. A device configured to decode video data, the device comprising: a memory configured to store the video data; and one or more processors configured to perform the methods of any of examples 1-18.

[0207] Example 38. A device configured to encode video data, the device comprising: a memory configured to store the video data; and one or more processors configured to perform the methods of any of examples 19-36.

[0208] Example 39. A device configured to decode video data, the device comprising means for performing the methods of any of examples 1-18.

[0209] Example 40. A device configured to encode video data, the device comprising means for performing the methods of any of examples 19-36.

[0210] Example 41. A computer-readable storage medium having instructions stored thereon that when executed cause one or more processors of a device to perform the methods of any of examples 1-18.

[0211] Example 42. A computer-readable storage medium having instructions stored thereon that when executed cause one or more processors of a device to perform the methods of any of examples 19-36.

[0212] It is to be recognized that depending on the example, certain acts or events of any of the techniques described herein can be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, acts or events may be performed concurrently, e.g., through multi-threaded processing, interrupt processing,

or multiple processors, rather than sequentially. In addition, while certain aspects of this disclosure are described as being performed by a single module or unit for purposes of clarity, it should be understood that the techniques of this disclosure may be performed by a combination of units or modules associated with a video coder.

[0213] Certain aspects of this disclosure have been described with respect to the HEVC standard for purposes of illustration. However, the techniques described in this disclosure may be useful for other video coding processes, including other standard or proprietary video coding processes not yet developed.

[0214] The techniques described above may be performed by video encoder 20 (FIGS. 1 and 5) and/or video decoder 30 (FIGS. 1 and 6), both of which may be generally referred to as a video coder. Likewise, video coding may refer to video encoding or video decoding, as applicable.

[0215] While particular combinations of various aspects of the techniques are described above, these combinations are provided merely to illustrate examples of the techniques described in this disclosure. Accordingly, the techniques of this disclosure should not be limited to these example combinations and may encompass any conceivable combination of the various aspects of the techniques described in this disclosure.

[0216] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over, as one or more instructions or code, a computer-readable medium 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.

[0217] By way of example, and not limitation, 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 transient media, but are instead directed to non-transient, 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.

[0218] 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 gate 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.

[0219] 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.

[0220] Various examples have been described. These and other examples are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method of decoding video data, the method comprising:

determining a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data;

including, in a palette for the current block, at least one candidate in the palette predictor list;

obtaining, based on one or more syntax elements signaled in a bitstream, an index value for a pixel of the current block;

identifying a palette entry in the palette that corresponds to the index value for the pixel; and

determining, based on the at least one sample value specified by the identified palette entry, at least one reconstructed sample value of the pixel.

2. The method of claim 1, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from at least one selected from the group consisting of: a line immediately above the current block, and a column immediately left of the current block.

3. The method of claim 2, wherein the reconstructed neighboring pixels include at least one selected from the group consisting of: a below-left reconstructed neighboring pixel, a left reconstructed neighboring pixel, an above-left reconstructed neighboring pixel, an above reconstructed neighboring pixel, or an above-right reconstructed neighboring pixel.

4. The method of claim 1, wherein determining the palette predictor list comprises:

removing one or more redundant candidates from the palette predictor list.

5. The method of claim 4, wherein determining the palette predictor list comprises:

determining, based on a metric of similarity of a first candidate in the palette predictor list and a second candidate in the palette predictor list, whether to remove the first candidate from the palette predictor list; and

removing the first candidate from the palette predictor list responsive to the determination.

6. The method of claim 1, wherein the one or more neighboring pixels include a plurality of neighboring pixels, and determining the palette predictor list comprises:

ranking, within the palette predictor list, two or more of the plurality of reconstructed neighboring pixels according to frequency of values of the plurality of the reconstructed neighboring pixels.

7. The method of claim 1, wherein determining the palette predictor list comprises:

truncating the palette predictor list such that a size of the palette predictor list is limited to a particular number of candidates.

8. The method of claim 1, further comprising applying in-loop filtering to the one or more reconstructed neighboring pixels after determining the palette predictor list.

9. The method of claim 1, comprising:

obtaining, from the bitstream, a binary vector comprising one or more syntax elements, wherein each of the one or more syntax elements corresponds to a different candidate in the palette predictor list;

for each respective candidate in the palette predictor list, determining, based on the binary vector syntax element corresponding to the respective candidate in the palette predictor list, the respective candidate in the palette predictor list is not included in the palette for the current block.

10. The method of claim 1, wherein the palette predictor list is a first palette predictor list and the binary vector is a first binary vector, the method further comprising:

determining a second palette predictor list comprising one or more candidates, each respective candidate in the second palette predictor list specifying a palette entry in a palette for a previously-coded block of the video data;

obtaining, from the bitstream, a binary vector for the second palette predictor list, the binary vector comprising one or more binary vector syntax elements, wherein each of the one or more binary vector syntax elements corresponds to a different candidate in the palette predictor list; and

for each respective candidate in the second palette predictor list, determining, based on the binary vector syntax element corresponding to the respective candidate in the second palette predictor list, the respective candidate in the second palette predictor list is not included in the palette for the current block.

11. The method of claim 10, further comprising:

applying a pruning process that removes, from the first palette predictor list, a particular candidate in the first palette predictor list if the particular candidate matches a candidate in the second palette predictor list.

12. The method of claim 10, further comprising:

determining, based on a metric of similarity of a first candidate in the first palette predictor list and a second candidate in the second palette predictor list, whether to remove the first candidate from the first palette predictor list; and

removing the first candidate from the first palette predictor list responsive to the determination.

13. The method of claim 10, further comprising:

obtaining, from the bitstream, syntax elements specifying one or more additional palette entries; and

including, in the palette for the current block, the one or more additional palette entries,

wherein the palette entries in the palette for the current block are ordered as follows: palette entries specifying candidates in the first palette predictor list, palette entries specifying candidates in the second palette predictor list, and the additional palette entries.

14. The method of claim 10, further comprising:

obtaining, from the bitstream, syntax elements specifying one or more additional palette entries; and

including, in the palette for the current block, the one or more additional palette entries,

wherein the palette entries in the palette for the current block are ordered as follows: palette entries specifying candidates in the second palette predictor list, palette entries specifying candidates in the first palette predictor list, and the additional palette entries.

15. The method of claim 1, wherein candidates of the palette predictor list are not included in a palette predictor list used to determine a palette for a block of the video data processed after the current block.

16. A method of encoding video data, the method comprising:

determining a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data;

including, in a palette for the current block, at least one candidate in the palette predictor list; and

signaling, in a bitstream, one or more syntax elements indicating an index value for a pixel of the current block, the index value corresponding to an entry in the palette.

17. The method of claim 16, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from at least one selected from the group consisting of: a line immediately above the current block, and a column immediately left of the current block.

18. The method of claim 16, wherein the reconstructed neighboring pixels include at least one selected from the group consisting of: a below-left reconstructed neighboring pixel, a left reconstructed neighboring pixel, an above-left reconstructed neighboring pixel, an above reconstructed neighboring pixel, or an above-right reconstructed neighboring pixel.

19. The method of claim 16, wherein the one or more neighboring pixels include a plurality of neighboring pixels, and generating the palette predictor list comprises:

ranking, within the palette predictor list, two or more of the plurality of reconstructed neighboring pixels according to frequency of values of the plurality of the reconstructed neighboring pixels.

20. The method of claim 16, further comprising:

signaling, in the bitstream, a binary vector comprising one or more syntax elements, wherein each of the one or more syntax elements corresponds to a different candidate in the palette predictor list; and

for each respective candidate in the palette predictor list, determining, based on the binary vector syntax element corresponding to the respective candidate in the palette predictor list, the respective candidate in the palette predictor list is not included in the palette for the current block.

21. The method of claim 16, wherein the palette predictor list is a first palette predictor list and the binary vector is a first binary vector, the method further comprising:

determining a second palette predictor list that comprises one or more candidates, each respective candidate in the second palette predictor list specifying a palette entry in a palette for a previously-coded block of the video data;

signaling a binary vector in the bitstream, the binary vector comprising one or more binary vector syntax elements, wherein each of the one or more binary vector syntax elements corresponds to a different candidate in the second palette predictor list; and

for at least one respective candidate in the second palette predictor list, determining, based on the binary vector syntax element corresponding to the respective candidate in the second palette predictor list, the respective candidate in the second palette predictor list is not included in the palette for the current block.

22. The method of claim 21, further comprising:

applying a pruning process that removes, from the first palette predictor list, a particular candidate in the first palette predictor list if the particular candidate matches a candidate in the second palette predictor list.

23. A device configured to code video data, the device comprising:

a memory configured to store the video data; and

one or more processors configured to:

determine a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data; and

include, in a palette for the current block, at least one candidate in the palette predictor list.

24. The device of claim 23, wherein the reconstructed neighboring pixels include reconstructed neighboring pixels from at least one selected from the group consisting of: line immediately above the current block, and a column immediately left of the current block.

25. The device of claim 23, the one or more processors are configured to:

obtain, from the bitstream, a binary vector comprising one or more syntax elements, wherein each of the one or more syntax elements corresponds to a different candidate in the palette predictor list; and

for each respective candidate in the palette predictor list, determine, based on the binary vector syntax element corresponding to the respective candidate in the palette predictor list, the respective candidate in the palette predictor list is not included in the palette for the current block.

26. The device of claim 23, wherein the palette predictor list is a first palette predictor list and the binary vector is a first binary vector, the one or more processors further configured to:

determine a second palette predictor list comprising one or more candidates, each respective candidate in the second palette predictor list specifying a palette entry in a palette for a previously-coded block of the video data;

obtain, from a bitstream, a binary vector for the second palette predictor list, the binary vector comprising one or more binary vector syntax elements, wherein each of the one or more binary vector syntax elements corresponds to a different candidate in the palette predictor list; and

for each respective candidate in the second palette predictor list, determine, based on the binary vector syntax element corresponding to the respective candidate in the second palette predictor list, the respective candidate in the second palette predictor list is not included in the palette for the current block.

27. The device of claim 23, wherein the one or more processors are configured to:

obtain, based on one or more additional syntax elements signaled in a bitstream, an index value for a pixel of the current block;

identify a palette entry in the palette that corresponds to the index value for the pixel; and

determine, based on the at least one sample value specified by the identified palette entry, at least one reconstructed sample value of the pixel.

28. The device of claim 23, wherein the device comprises at least one selected from the group consisting of:

an integrated circuit;

a microprocessor; or

a wireless communication device.

29. The device of claim 23, further comprising at least one selected from the group consisting of: a display configured to display the decoded video data, or a camera configured to capture the video data.

30. A device configured to decode video data, the device comprising:

means for determining a palette predictor list comprising one or more candidates, each respective candidate in the palette predictor list specifying at least one sample value of a different respective reconstructed neighboring pixel from among one or more reconstructed neighboring pixels, wherein each of the one or more reconstructed neighboring pixels is in a line above or a column left of a current block of the video data; and

means for including, in a palette for the current block, at least one candidate in the palette predictor list.

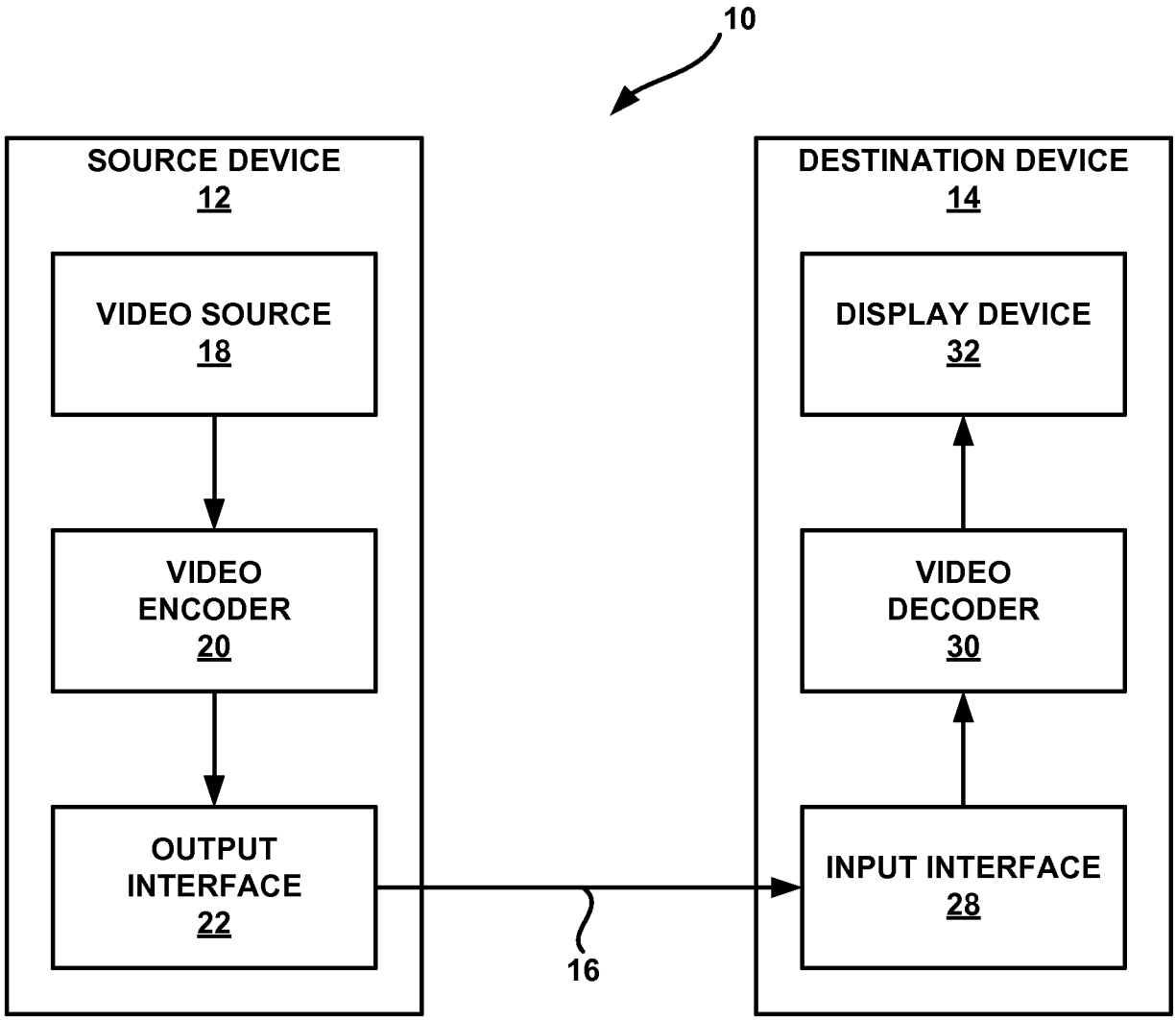
10
SOURCE DEVICE
12
VIDEO SOURCE
18
VIDEO
ENCODER
20
OUTPUT
INTERFACE
22
16
DESTINATION DEVICE
14
DISPLAY DEVICE
32
VIDEO
DECODER
30
INPUT INTERFACE
28

FIG. 1

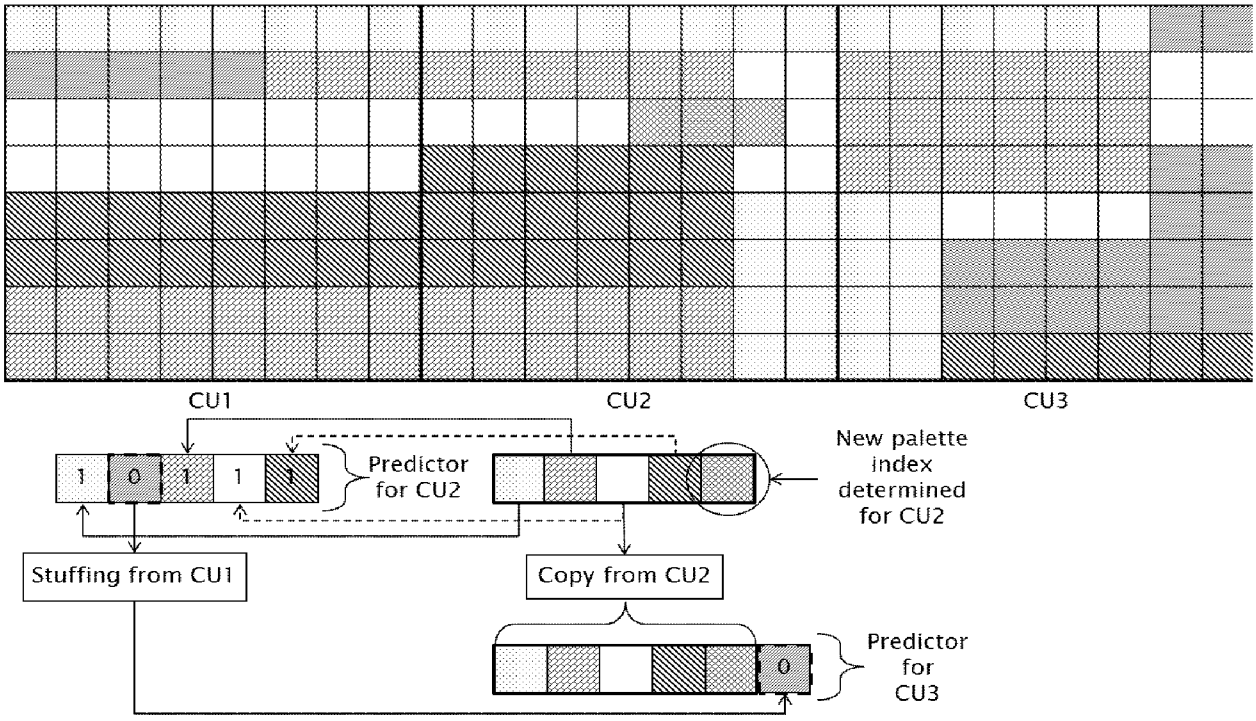
CU1
CU2
CU3
1
0
1
1
1
Predictor for CU2
New palette index determined for CU2
Stuffing from CU1
Copy from CU2
0
Predictor for CU3

FIG. 2

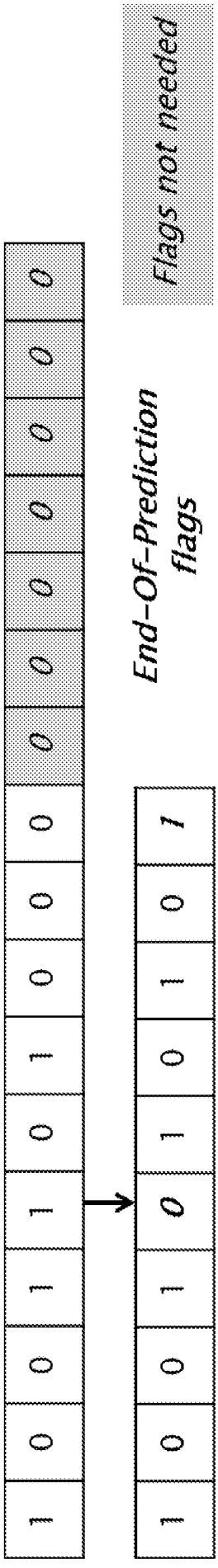
Flags not needed
End-Of-Prediction flags
1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 0 1 0 1 0 1

FIG. 3

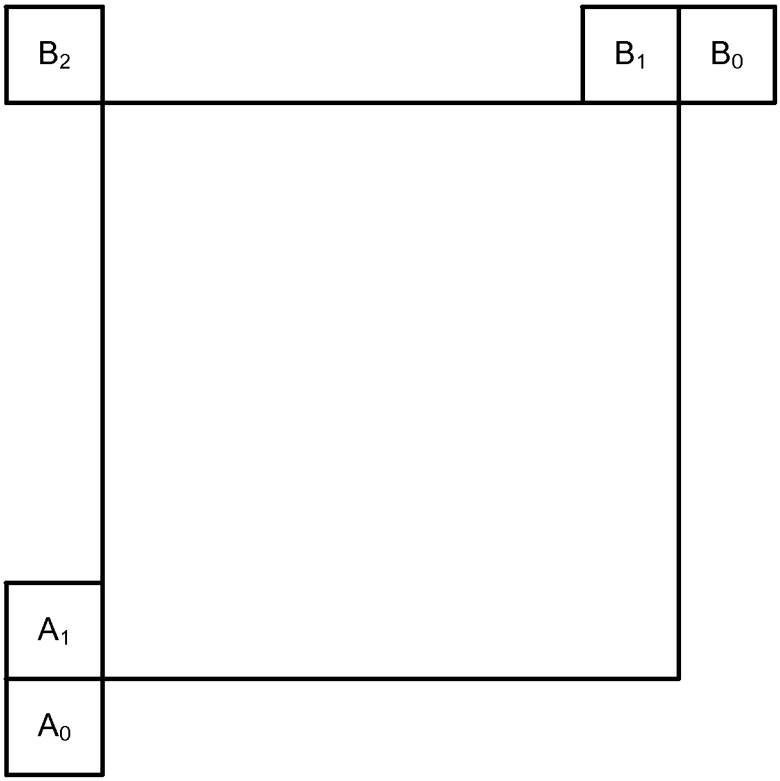

FIG. 4

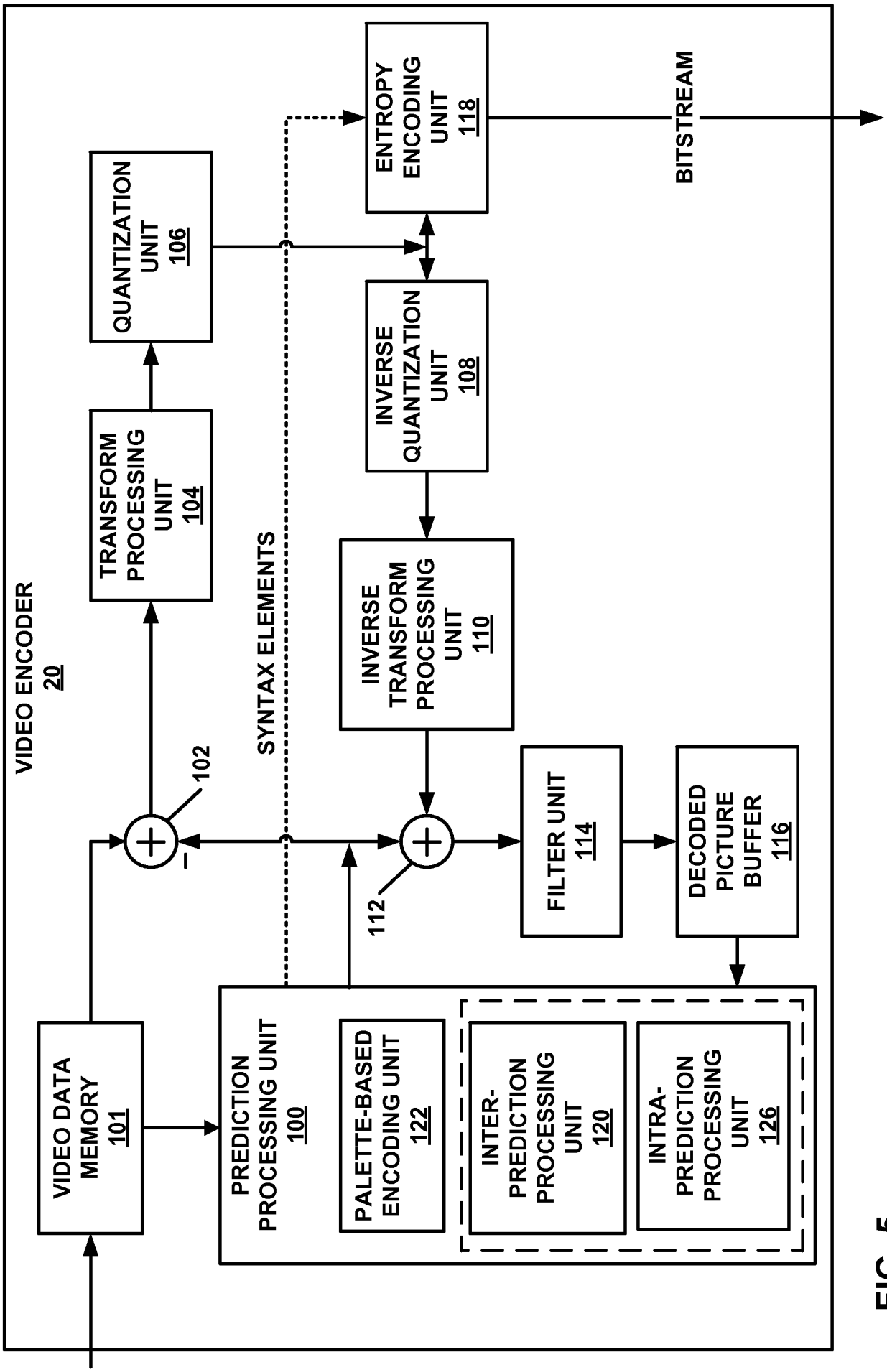
VIDEO ENCODER
20
VIDEO DATA MEMORY
101
PREDICTION PROCESSING UNIT
100
PALETTE-BASED ENCODING UNIT
122
INTER-PREDICTION PROCESSING UNIT
120
INTRA-PREDICTION PROCESSING UNIT
126
TRANSFORM PROCESSING UNIT
104
QUANTIZATION UNIT
106
INVERSE QUANTIZATION UNIT
108
INVERSE TRANSFORM PROCESSING UNIT
110
ENTROPY ENCODING UNIT
118
BITSTREAM
SYNTAX ELEMENTS
102
112
FILTER UNIT
114
DECODED PICTURE BUFFER
116

FIG. 5

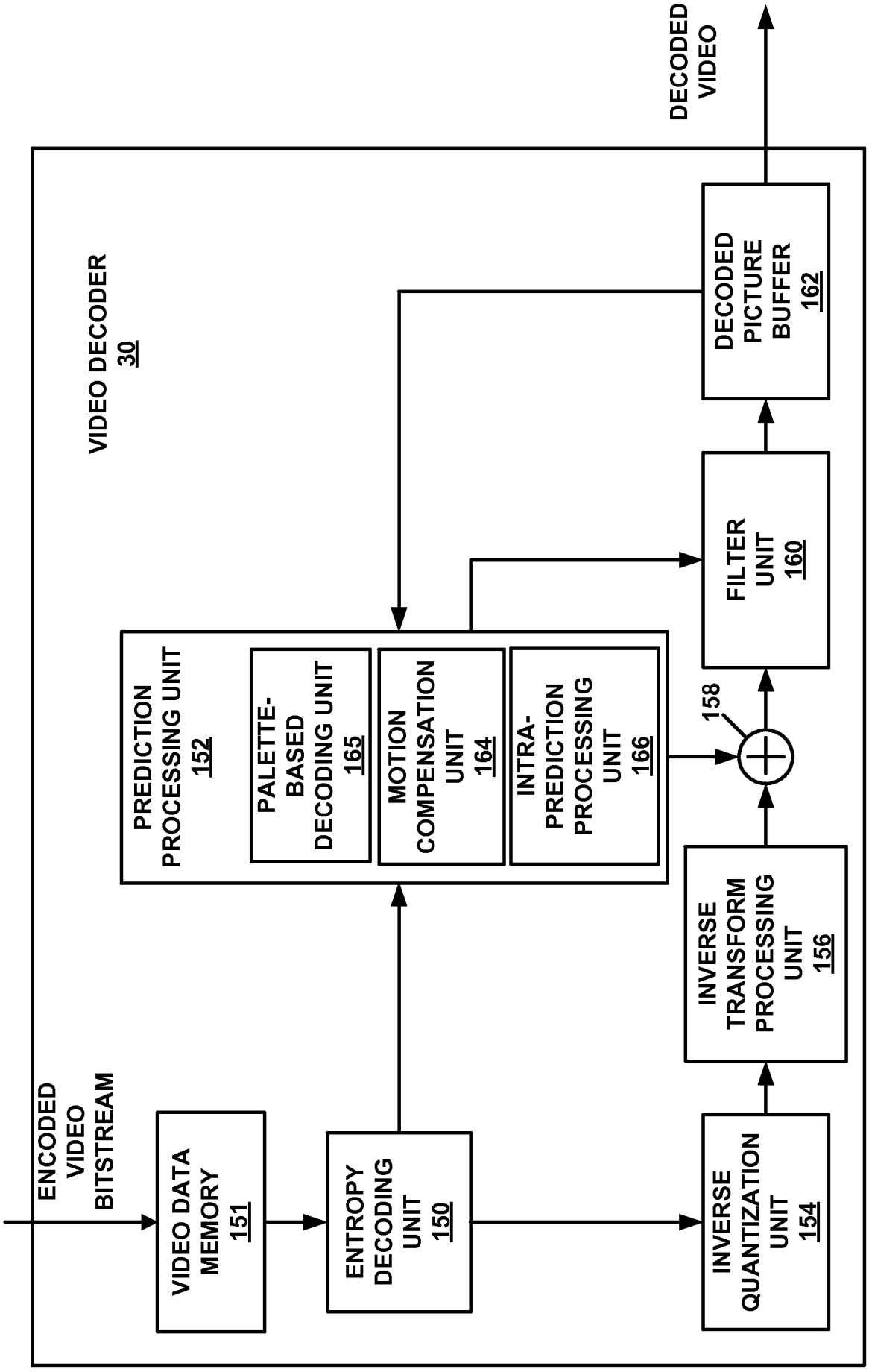
ENCODED VIDEO BITSTREAM
VIDEO DECODER 30
VIDEO DATA MEMORY 151
ENTROPY DECODING UNIT 150
PREDICTION PROCESSING UNIT 152
PALETTE-BASED DECODING UNIT 165
MOTION COMPENSATION UNIT 164
INTRA-PREDICTION PROCESSING UNIT 166
INVERSE QUANTIZATION UNIT 154
INVERSE TRANSFORM PROCESSING UNIT 156
158
FILTER UNIT 160
DECODED PICTURE BUFFER 162
DECODED VIDEO

FIG. 6

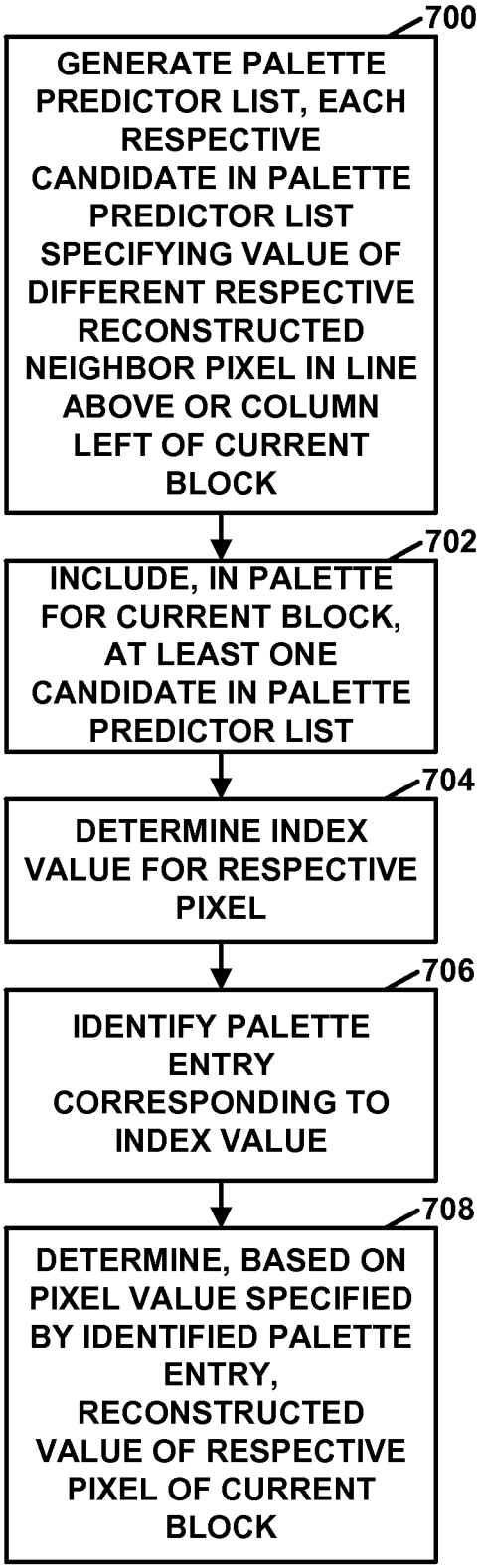
700
GENERATE PALETTE PREDICTOR LIST, EACH RESPECTIVE CANDIDATE IN PALETTE PREDICTOR LIST SPECIFYING VALUE OF DIFFERENT RESPECTIVE RECONSTRUCTED NEIGHBOR PIXEL IN LINE ABOVE OR COLUMN LEFT OF CURRENT BLOCK
702
INCLUDE, IN PALETTE FOR CURRENT BLOCK, AT LEAST ONE CANDIDATE IN PALETTE PREDICTOR LIST
704
DETERMINE INDEX VALUE FOR RESPECTIVE PIXEL
706
IDENTIFY PALETTE ENTRY CORRESPONDING TO INDEX VALUE
708
DETERMINE, BASED ON PIXEL VALUE SPECIFIED BY IDENTIFIED PALETTE ENTRY, RECONSTRUCTED VALUE OF RESPECTIVE PIXEL OF CURRENT BLOCK

FIG. 7

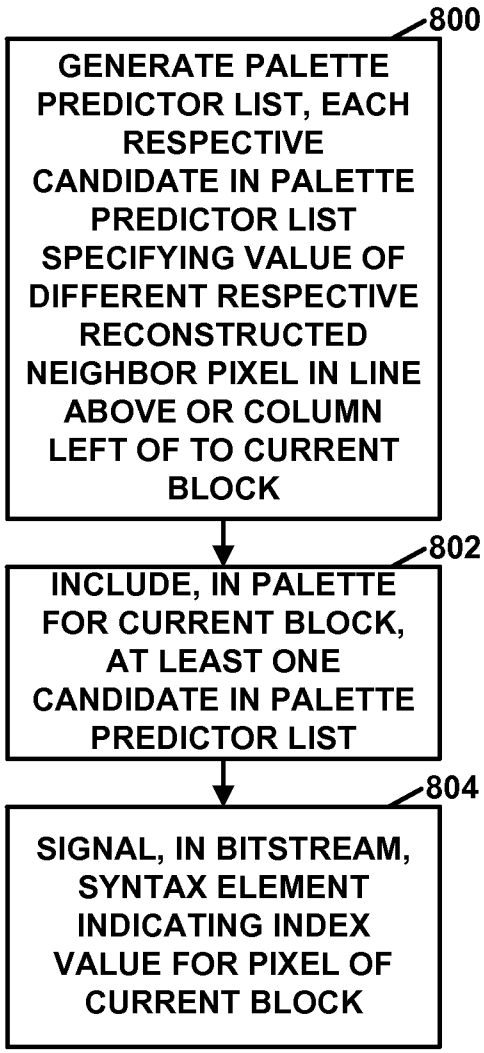
800
GENERATE PALETTE PREDICTOR LIST, EACH RESPECTIVE CANDIDATE IN PALETTE PREDICTOR LIST SPECIFYING VALUE OF DIFFERENT RESPECTIVE RECONSTRUCTED NEIGHBOR PIXEL IN LINE ABOVE OR COLUMN LEFT OF TO CURRENT BLOCK
802
INCLUDE, IN PALETTE FOR CURRENT BLOCK, AT LEAST ONE CANDIDATE IN PALETTE PREDICTOR LIST
804
SIGNAL, IN BITSTREAM, SYNTAX ELEMENT INDICATING INDEX VALUE FOR PIXEL OF CURRENT BLOCK

FIG. 8

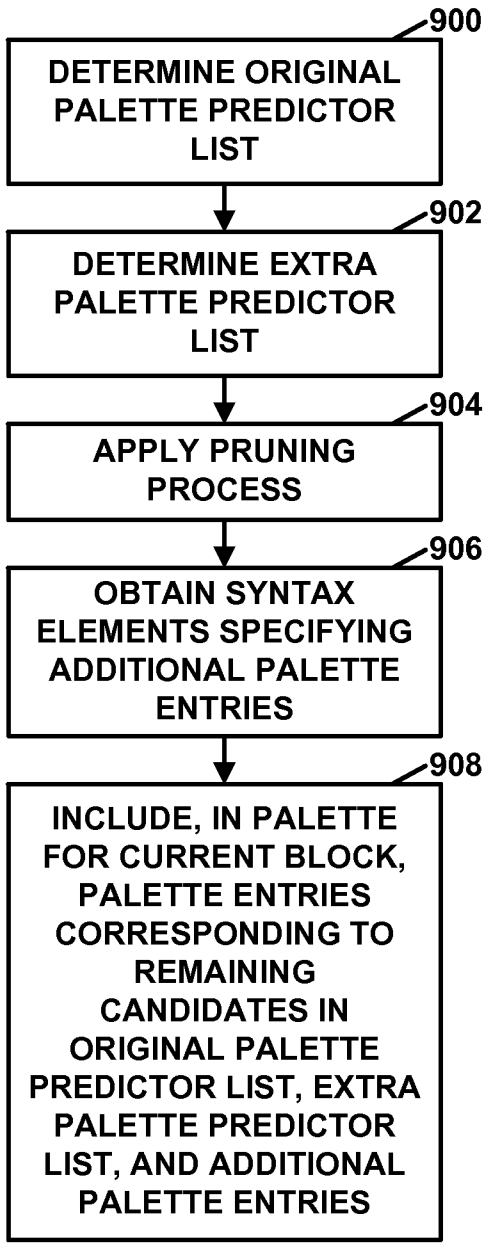
900
DETERMINE ORIGINAL PALETTE PREDICTOR LIST
902
DETERMINE EXTRA PALETTE PREDICTOR LIST
904
APPLY PRUNING PROCESS
906
OBTAIN SYNTAX ELEMENTS SPECIFYING ADDITIONAL PALETTE ENTRIES
908
INCLUDE, IN PALETTE FOR CURRENT BLOCK, PALETTE ENTRIES CORRESPONDING TO REMAINING CANDIDATES IN ORIGINAL PALETTE PREDICTOR LIST, EXTRA PALETTE PREDICTOR LIST, AND ADDITIONAL PALETTE ENTRIES

FIG. 9

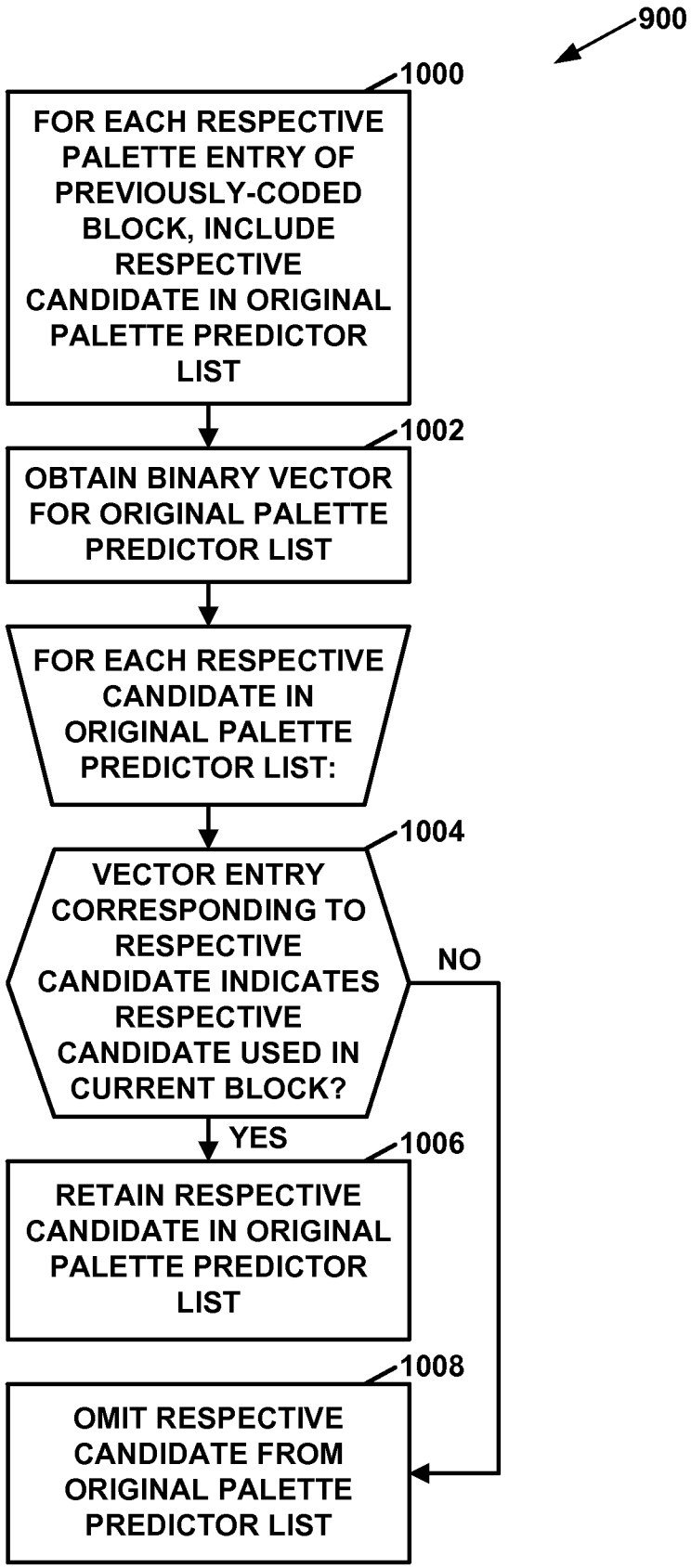
900
1000
FOR EACH RESPECTIVE PALETTE ENTRY OF PREVIOUSLY-CODED BLOCK, INCLUDE RESPECTIVE CANDIDATE IN ORIGINAL PALETTE PREDICTOR LIST
1002
OBTAIN BINARY VECTOR FOR ORIGINAL PALETTE PREDICTOR LIST
FOR EACH RESPECTIVE CANDIDATE IN ORIGINAL PALETTE PREDICTOR LIST:
1004
VECTOR ENTRY CORRESPONDING TO RESPECTIVE CANDIDATE INDICATES RESPECTIVE CANDIDATE USED IN CURRENT BLOCK?
NO
YES
1006
RETAIN RESPECTIVE CANDIDATE IN ORIGINAL PALETTE PREDICTOR LIST
1008
OMIT RESPECTIVE CANDIDATE FROM ORIGINAL PALETTE PREDICTOR LIST

FIG. 10

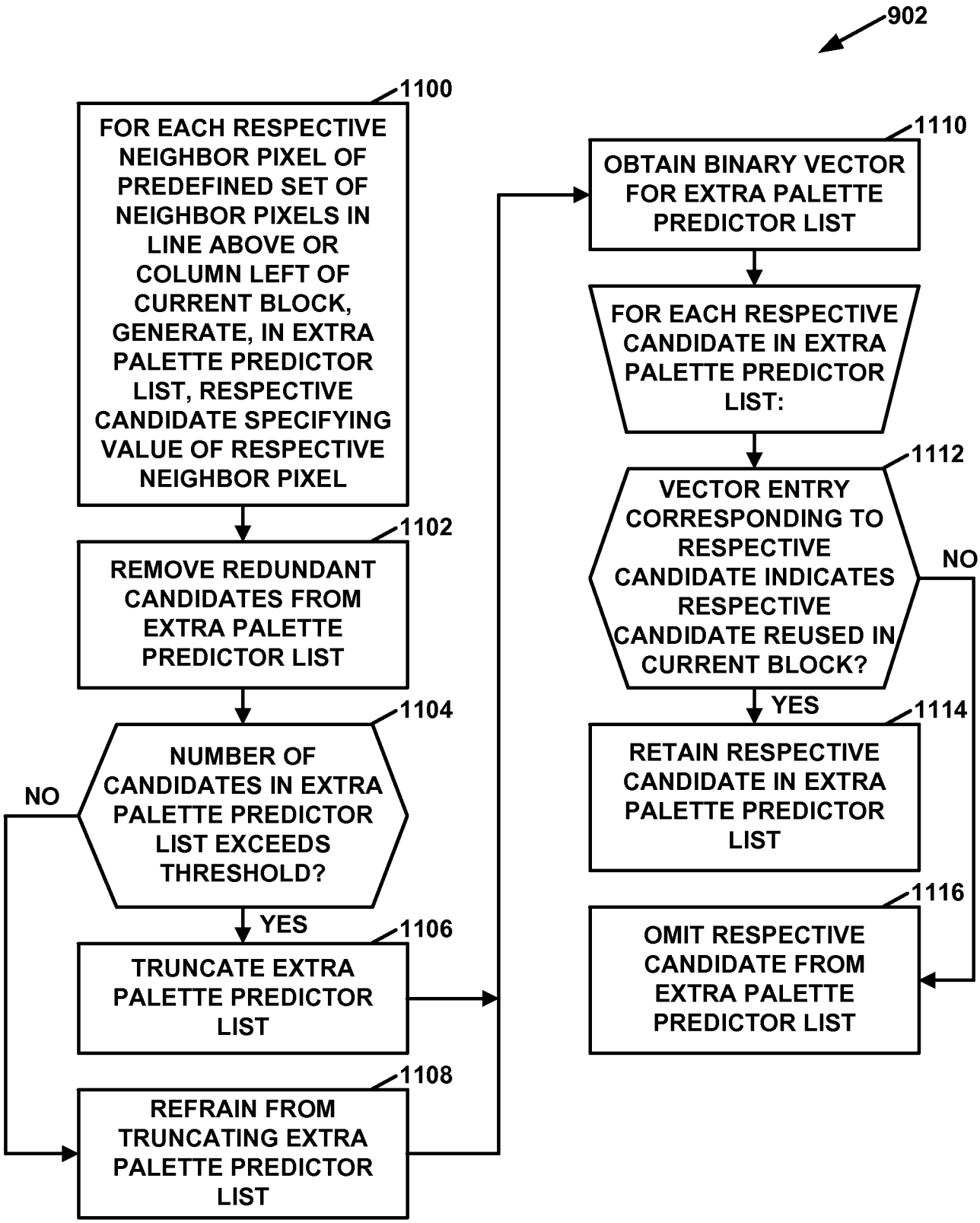
902
1100
FOR EACH RESPECTIVE NEIGHBOR PIXEL OF PREDEFINED SET OF NEIGHBOR PIXELS IN LINE ABOVE OR COLUMN LEFT OF CURRENT BLOCK, GENERATE, IN EXTRA PALETTE PREDICTOR LIST, RESPECTIVE CANDIDATE SPECIFYING VALUE OF RESPECTIVE NEIGHBOR PIXEL
1102
REMOVE REDUNDANT CANDIDATES FROM EXTRA PALETTE PREDICTOR LIST
1104
NUMBER OF CANDIDATES IN EXTRA PALETTE PREDICTOR LIST EXCEEDS THRESHOLD?
NO
YES
1106
TRUNCATE EXTRA PALETTE PREDICTOR LIST
1108
REFRAIN FROM TRUNCATING EXTRA PALETTE PREDICTOR LIST
1110
OBTAIN BINARY VECTOR FOR EXTRA PALETTE PREDICTOR LIST
FOR EACH RESPECTIVE CANDIDATE IN EXTRA PALETTE PREDICTOR LIST:
1112
VECTOR ENTRY CORRESPONDING TO RESPECTIVE CANDIDATE INDICATES RESPECTIVE CANDIDATE REUSED IN CURRENT BLOCK?
NO
YES
1114
RETAIN RESPECTIVE CANDIDATE IN EXTRA PALETTE PREDICTOR LIST
1116
OMIT RESPECTIVE CANDIDATE FROM EXTRA PALETTE PREDICTOR LIST

FIG. 11

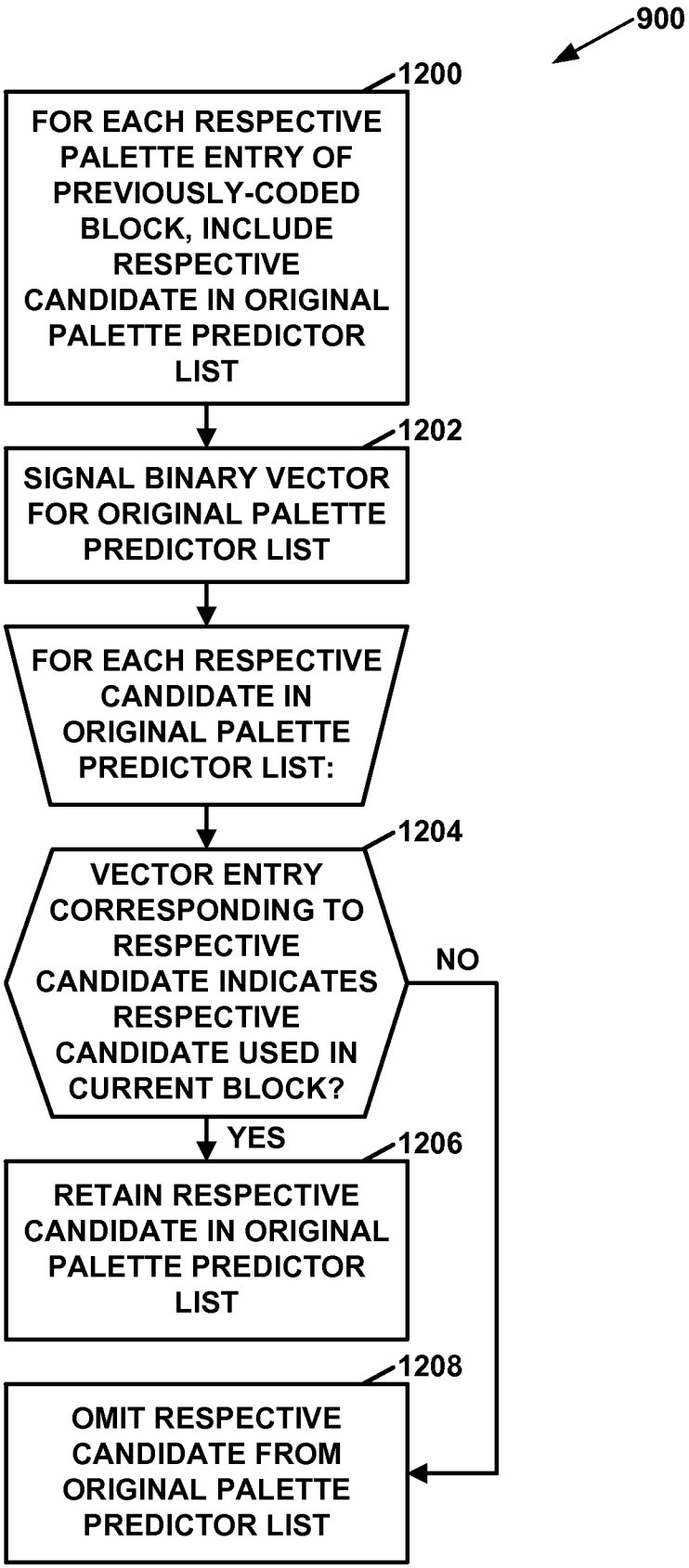
900
1200
FOR EACH RESPECTIVE PALETTE ENTRY OF PREVIOUSLY-CODED BLOCK, INCLUDE RESPECTIVE CANDIDATE IN ORIGINAL PALETTE PREDICTOR LIST
1202
SIGNAL BINARY VECTOR FOR ORIGINAL PALETTE PREDICTOR LIST
FOR EACH RESPECTIVE CANDIDATE IN ORIGINAL PALETTE PREDICTOR LIST:
1204
VECTOR ENTRY CORRESPONDING TO RESPECTIVE CANDIDATE INDICATES RESPECTIVE CANDIDATE USED IN CURRENT BLOCK?
NO
YES
1206
RETAIN RESPECTIVE CANDIDATE IN ORIGINAL PALETTE PREDICTOR LIST
1208
OMIT RESPECTIVE CANDIDATE FROM ORIGINAL PALETTE PREDICTOR LIST

FIG. 12

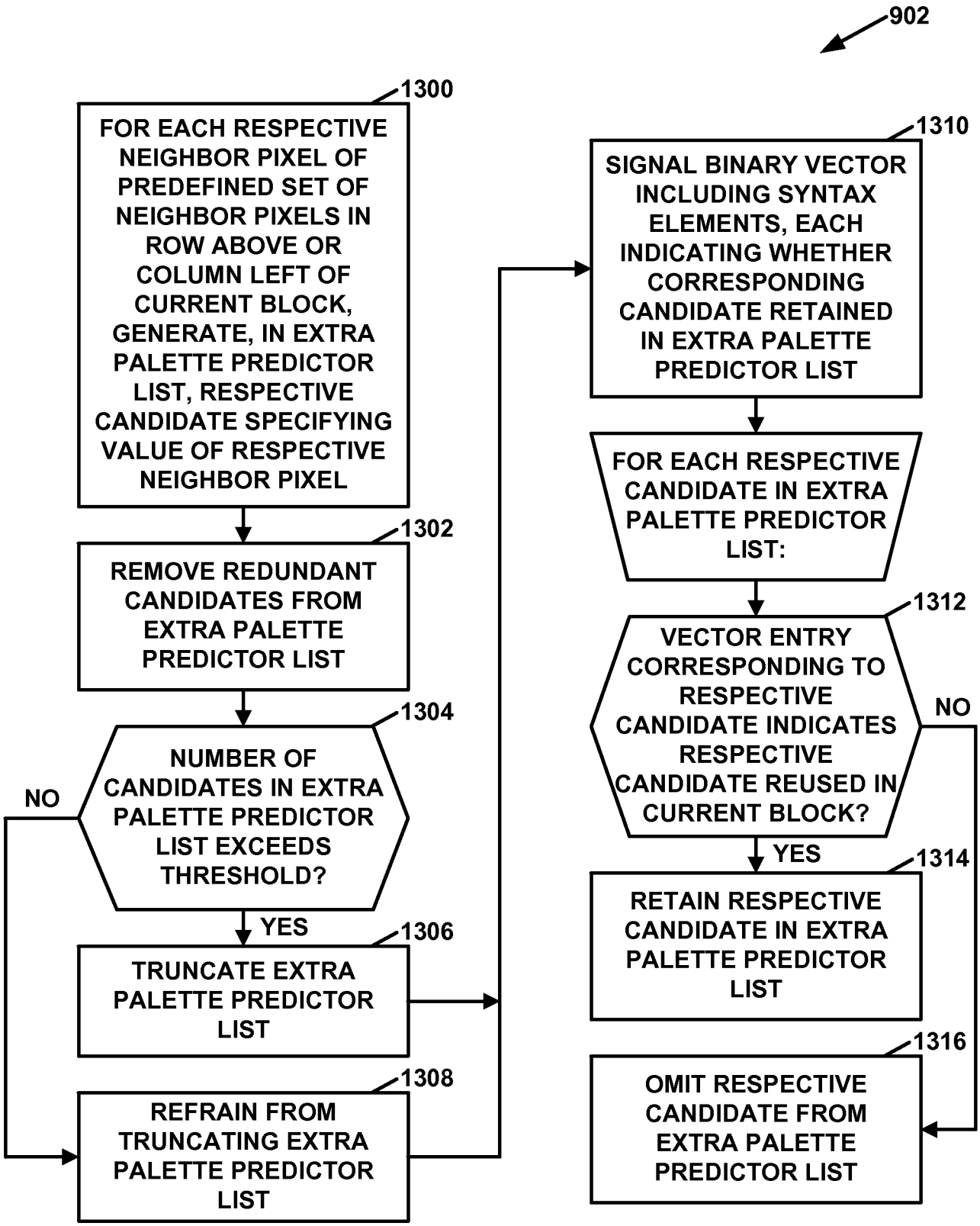
902
1300
FOR EACH RESPECTIVE NEIGHBOR PIXEL OF PREDEFINED SET OF NEIGHBOR PIXELS IN ROW ABOVE OR COLUMN LEFT OF CURRENT BLOCK, GENERATE, IN EXTRA PALETTE PREDICTOR LIST, RESPECTIVE CANDIDATE SPECIFYING VALUE OF RESPECTIVE NEIGHBOR PIXEL
1302
REMOVE REDUNDANT CANDIDATES FROM EXTRA PALETTE PREDICTOR LIST
1304
NUMBER OF CANDIDATES IN EXTRA PALETTE PREDICTOR LIST EXCEEDS THRESHOLD?
NO
YES
1306
TRUNCATE EXTRA PALETTE PREDICTOR LIST
1308
REFRAIN FROM TRUNCATING EXTRA PALETTE PREDICTOR LIST
1310
SIGNAL BINARY VECTOR INCLUDING SYNTAX ELEMENTS, EACH INDICATING WHETHER CORRESPONDING CANDIDATE RETAINED IN EXTRA PALETTE PREDICTOR LIST
FOR EACH RESPECTIVE CANDIDATE IN EXTRA PALETTE PREDICTOR LIST:
1312
VECTOR ENTRY CORRESPONDING TO RESPECTIVE CANDIDATE INDICATES RESPECTIVE CANDIDATE REUSED IN CURRENT BLOCK?
NO
YES
1314
RETAIN RESPECTIVE CANDIDATE IN EXTRA PALETTE PREDICTOR LIST
1316
OMIT RESPECTIVE CANDIDATE FROM EXTRA PALETTE PREDICTOR LIST

FIG. 13

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/039904

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/70 H04N19/90 H04N19/593
ADD.

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)
H04N

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)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	XIU X ET AL: "Description of screen content coding technology proposal by InterDigital", 17. JCT-VC MEETING; 27-3-2014 - 4-4-2014; VALENCIA; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://WFTP3.ITU.INT/AV-ARCH/JCTVC-SITE/,, no. JCTVC-Q0037, 18 March 2014 (2014-03-18), XP030115927,	1-9, 15-20, 23-25, 27-30
Y	paragraph [2.6.1]	10-14, 21,22,26

-/--

[X] 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
29 September 2015	07/10/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Oelbaum, Tobias

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/039904

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ZHU J ET AL: "Non-SCCE3: Modified Escaped pixel mode in palette based coding", 18. JCT-VC MEETING; 30-6-2014 - 9-7-2014; SAPPORO; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://WFTP3.ITU.INT/AV-ARCH/JCTVC-SITE/,, no. JCTVC-R0080-v2, 2 July 2014 (2014-07-02), XP030116331,	1,16,23, 30
Y	paragraph [02.1]	10-14, 21,22,26

X	JIN G ET AL: "Non-RCE4: Palette prediction for palette coding", 16. JCT-VC MEETING; 9-1-2014 - 17-1-2014; SAN JOSE; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://WFTP3.ITU.INT/AV-ARCH/JCTVC-SITE/,, no. JCTVC-P0160, 4 January 2014 (2014-01-04), XP030115679,	1,16,23, 30
Y	abstract	13,14

X	GISQUET C ET AL: "SCCE3: Test A.3 - Palette stuffing", 18. JCT-VC MEETING; 30-6-2014 - 9-7-2014; SAPPORO; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://WFTP3.ITU.INT/AV-ARCH/JCTVC-SITE/,, no. JCTVC-R0082, 20 June 2014 (2014-06-20) , XP030116334,	1,16,23, 30
Y	abstract	13,14

1