



- (51) International Patent Classification:
H04N 19/176 (2014.01)

(21) International Application Number:
PCT/US2024/024312

(22) International Filing Date:
12 April 2024 (12.04.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/495,677 12 April 2023 (12.04.2023) US

(71) Applicant: **BEIJING DAJIA INTERNET INFORMATION TECHNOLOGY CO., LTD** [CN/CN]; Room 101, 8th Floor, Building 12, No. 16, Xierqi West Road, Haidian District, Beijing 100085 (CN).
- (71) Applicant (*for MN only*): **JHU, Hong-Jheng** [CN/US]; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US).

(72) Inventors: **JHU, Hong-Jheng**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **KUO, Che-Wei**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **XIU, Xiaoyu**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **YAN, Ning**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **CHEN, Wei**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **MA, Changyue**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **WANG, Xianglin**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **YU, Bing**; No. 29 Xi'erqi Middle Road, Qinghe Street, Haidian District, Beijing (CN).

(54) Title: METHODS AND DEVICES FOR FILTERED INTRA BLOCK COPY

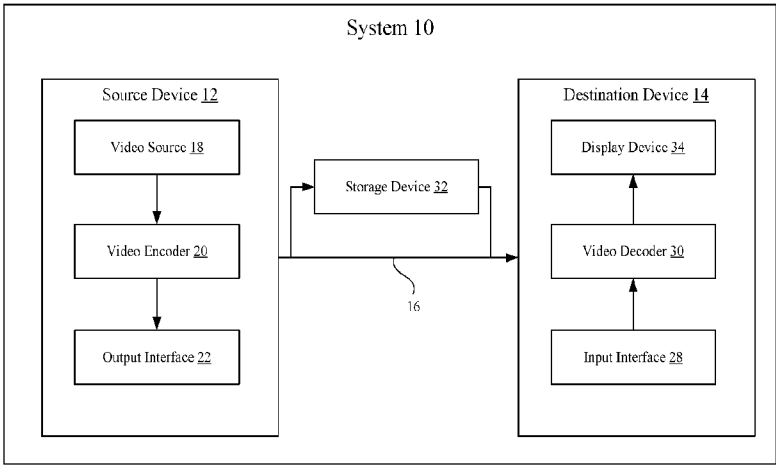


Figure 1

(57) Abstract: The present disclosure provides a method for video decoding, comprising: determining a reference block in a video frame from a bitstream for predicting a current block in the video frame; obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a template area associated with the reference block and a template area associated with the current block, wherein the template area associated with the reference block is extended based on the filter shape; deriving, with the set of filter coefficients and the filter shape, each of predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block; and reconstructing the current block based on the predicted sample values.



(74) **Agent: LOVE, James et al.;** Oblon, McClelland, Maier & Neustadt, L.L.P., 1940 Duke Street, Alexandria, Virginia 22314 (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, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, 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, ME, 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:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

METHODS AND DEVICES FOR FILTERED INTRA BLOCK COPY

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is based upon and claims priority to Provisional Application No. 63/495,677 filed on April 12, 2023. The entire content thereof is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] This application is related to video coding and compression. More specifically, this application relates to methods and apparatus on improving the coding efficiency of filtered intra block copy (FIBC).

BACKGROUND

[0003] Digital video is supported by a variety of electronic devices, such as digital televisions, laptop or desktop computers, tablet computers, digital cameras, digital recording devices, digital media players, video gaming consoles, smart phones, video teleconferencing devices, video streaming devices, etc. The electronic devices transmit and receive or otherwise communicate digital video data across a communication network, and/or store the digital video data on a storage device. Due to a limited bandwidth capacity of the communication network and limited memory resources of the storage device, video coding may be used to compress the video data according to one or more video coding standards before it is communicated or stored. For example, video coding standards include Versatile Video Coding (VVC), Joint Exploration test Model (JEM), High-Efficiency Video Coding (HEVC/H.265), Advanced Video Coding (AVC/H.264), Moving Picture Expert Group (MPEG) coding, or the like. Video coding generally utilizes prediction methods (e.g., inter-prediction, intra-prediction, or the like) that take advantage of redundancy inherent in the video data. Video coding aims to compress video data into a form that uses a lower bit rate, while avoiding or minimizing degradations to video quality.

SUMMARY

[0004] Embodiments of the present disclosure provide methods and apparatus on improving the coding efficiency of the image/video blocks which applies FIBC technology.

[0005] According to one aspect of the present disclosure, there is provided a method for video decoding, comprising: determining a reference block in a video frame from a bitstream for predicting a current block in the video frame; obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a template area associated

with the reference block and a template area associated with the current block, wherein the template area associated with the reference block is extended based on the filter shape; deriving, with the set of filter coefficients and the filter shape, each of predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block; and reconstructing the current block based on the predicted sample values.

[0006] According to one aspect of the present disclosure, there is provided a method for video encoding, comprising: determining a reference block in a video frame for predicting a current block in the video frame; obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a template area associated with the reference block and a template area associated with the current block, wherein the template area associated with the reference block is extended based on the filter shape; deriving, with the set of filter coefficients and the filter shape, each of predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block; and generating a bitstream based on the predicted sample values.

[0007] According to one aspect of the present disclosure, there is provided a method for video decoding, comprising: obtaining a merge candidate list for intra block copy (IBC) prediction of a current block, the merge candidate list comprises a plurality of candidates that have been encoded with IBC; reordering the plurality of candidates in the merge candidate list based on a template matching score of each of the plurality of candidates, the template matching score is obtained based on a difference between sample values of a template of a candidate and corresponding reference sample values of a reference template of a reference block of that candidate, wherein in response to determining that a candidate is coded with filtered IBC, the corresponding reference sample values of the reference template are obtained using filtered IBC; and reconstructing the current block based on the reordered merge candidate list.

[0008] According to one aspect of the present disclosure, there is provided a method for video encoding, comprising: obtaining a merge candidate list for intra block copy (IBC) prediction of a current block, the merge candidate list comprises a plurality of candidates that have been encoded with IBC; reordering the plurality of candidates in the merge candidate list based on a template matching score of each of the plurality of candidates, the template matching score is obtained based on a difference between sample values of a template of a candidate and corresponding reference sample values of a reference template of a reference block of that candidate, wherein in response to determining that a candidate is coded with filtered IBC, the corresponding reference sample values of the reference template are obtained using filtered IBC; and generating a bitstream by encoding the current block based on the reordered merge candidate list.

[0009] According to one aspect of the present disclosure, there is provided a method for video decoding, comprising: determining a block vector for a chroma block from a bitstream based on a luma block that is coded with an intra block copy (IBC) and is associated with the

chroma block; obtaining an IBC prediction of the chroma block based on the block vector; in response to determining that a filtered IBC prediction is used for the chroma block, filtering the IBC prediction of the chroma block to obtain the filtered IBC prediction; and reconstructing the chroma block based on the filtered IBC prediction.

[0010] According to one aspect of the present disclosure, there is provided a method for video encoding, comprising: determining a block vector for a chroma block based on a luma block that is coded with an intra block copy (IBC) and is associated with the chroma block; obtaining an IBC prediction of the chroma block based on the block vector; filtering the IBC prediction of the chroma block; and generating a bitstream based on the filtered IBC prediction.

[0011] According to one aspect of the present disclosure, there is provided an apparatus, comprising: one or more processors; and one or more storage devices storing computer-executable instructions that, when executed, cause the one or more processors to perform the operations of the method of the present disclosure.

[0012] According to one aspect of the present disclosure, there is provided a computer program product, storing computer-executable instructions that, when executed, cause one or more processors to perform the operations of the method of the present disclosure.

[0013] According to one aspect of the present disclosure, there is provided a computer readable storage medium storing instructions which when executed by a computing device having one or more processors, cause the one or more processors to perform the decoding method of the present disclosure and storing a bitstream to be decoded by the decoding method of the present disclosure, or perform the encoding method of the present disclosure and storing a bitstream generated by the encoding method of the present disclosure.

[0014] According to one aspect of the present disclosure, there is provided a computer readable medium storing a bitstream, wherein the bitstream is to be decoded by performing the operations of the method of the present disclosure, or the bitstream is obtained by performing the operations of the method of the present disclosure.

[0015] According to one aspect of the present disclosure, there is provided a method for receiving a bitstream to be decoded by the decoding method of the present disclosure.

[0016] According to one aspect of the present disclosure, there is provided a method for transmitting a bitstream generated by the encoding method of the present disclosure.

[0017] It is to be understood that both the foregoing general description and the following detailed description are examples only and are not restrictive of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate examples consistent with the present disclosure and, together with the description, serve to explain the principles of the disclosure.

[0019] Figure 1 is a block diagram illustrating an exemplary system for encoding and

decoding video blocks in accordance with some implementations of the present disclosure.

[0020] Figure 2 is a block diagram illustrating an exemplary video encoder in accordance with some implementations of the present disclosure.

[0021] Figure 3 is a block diagram illustrating an exemplary video decoder in accordance with some implementations of the present disclosure.

[0022] Figures 4A, 4B, 4C, 4D, and 4E are block diagrams illustrating how a frame is recursively partitioned into multiple video blocks of different sizes and shapes in accordance with some implementations of the present disclosure.

[0023] Figure 5 illustrates a diagram of positions of spatial candidates.

[0024] Figure 6 illustrates a diagram of candidate pairs considered for redundancy check of spatial candidates.

[0025] Figure 7 illustrates a diagram of scaling of a motion vector for a temporal candidate.

[0026] Figure 8 illustrates a diagram of candidate positions for a temporal candidate.

[0027] Figure 9 illustrates a diagram of Merge mode with Motion Vector Difference (MMVD) search points.

[0028] Figure 10 illustrates uni-prediction motion vector selection for Geometric Partitioning Mode (GPM).

[0029] Figure 11 illustrates top and left neighboring blocks used in CIIP weight derivation.

[0030] Figure 12 illustrates current CTU processing order and its available reference samples in current and left CTU.

[0031] Figure 13 illustrates padding candidates for the replacement of the zero-vector in the IBC list.

[0032] Figure 14 illustrates reference area for IBC when CTU (m,n) is coded.

[0033] Figure 15 illustrates IBC reference area for camera-captured content.

[0034] Figures 16A to 16B illustrate the division method for angular modes.

[0035] Figures 17A to 17C illustrate available IPM candidates and Figure 17D illustrates an example of GPM with intra and intra prediction.

[0036] Figure 18 illustrates the edge on templates.

[0037] Figure 19 illustrates the intra template matching search area used.

[0038] Figure 20 illustrates the template used for template matching based OBMC.

[0039] Figure 21 illustrates the template and reference samples of the template in reference pictures.

[0040] Figure 22 illustrates the template and reference samples of the template for block with sub-block motion using the motion information of the subblocks of the current block.

[0041] Figure 23 illustrates the luma blocks used to derive direct block vector.

[0042] Figure 24 illustrates the division method and corresponding weights for intra coded block for angular and planar modes.

[0043] Figure 25 illustrates a diagram of the filter shape and training area of the reference

block.

[0044] Figure 26 illustrates a diagram of spatial terms correspond to neighboring luma samples.

[0045] Figure 27 illustrates a diagram of examples of different shape/number of filter taps.

[0046] Figure 28 illustrates a diagram of possible positions of candidate regions.

[0047] Figure 29 illustrates a diagram of possible positions of candidates.

[0048] Figure 30 illustrates a workflow of a method for decoding video data according to one or more aspects of the present disclosure.

[0049] Figure 31 illustrates a workflow of a method for encoding video data according to one or more aspects of the present disclosure.

[0050] Figure 32 illustrates a workflow of a method for decoding video data according to one or more aspects of the present disclosure.

[0051] Figure 33 illustrates a workflow of a method for encoding video data according to one or more aspects of the present disclosure.

[0052] Figure 34 illustrates a diagram of the filter shape and training area of the reference block.

[0053] Figure 35 illustrates a workflow of a method for video decoding according to one or more aspects of the present disclosure.

[0054] Figure 36 illustrates a workflow of a method for video encoding according to one or more aspects of the present disclosure.

[0055] Figure 37 illustrates a workflow of a method for video decoding according to one or more aspects of the present disclosure.

[0056] Figure 38 illustrates a workflow of a method for video encoding according to one or more aspects of the present disclosure.

[0057] Figure 39 illustrates a workflow of a method for video decoding according to one or more aspects of the present disclosure.

[0058] Figure 40 illustrates a workflow of a method for video encoding according to one or more aspects of the present disclosure.

[0059] Figure 41 is a diagram illustrating a computing environment coupled with a user interface, according to some implementations of the present disclosure.

DETAILED DESCRIPTION

[0060] Reference will now be made in detail to specific implementations, examples of which are illustrated in the accompanying drawings. In the following detailed description, numerous non-limiting specific details are set forth in order to assist in understanding the subject matter presented herein. But various alternatives may be used without departing from the scope of claims and the subject matter may be practiced without these specific details. For example, the subject matter presented herein can be implemented on many types of electronic

devices with digital video capabilities.

[0061] It should be illustrated that the terms “first,” “second,” and the like used in the description, claims of the present disclosure, and the accompanying drawings are used to distinguish objects, and not used to describe any specific order or sequence. It should be understood that the data used in this way may be interchanged under an appropriate condition, such that the embodiments of the present disclosure described herein may be implemented in orders besides those shown in the accompanying drawings or described in the present disclosure.

[0062] Figure 1 is a block diagram illustrating an exemplary system 10 for encoding and decoding video blocks in parallel in accordance with some implementations of the present disclosure. As shown in Figure 1, the system 10 includes a source device 12 that generates and encodes video data to be decoded at a later time by a destination device 14. The source device 12 and the destination device 14 may comprise any of a wide variety of electronic devices, including cloud servers, server computers, desktop or laptop computers, tablet computers, smart phones, set-top boxes, digital televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some implementations, the source device 12 and the destination device 14 are equipped with wireless communication capabilities.

[0063] In some implementations, the destination device 14 may receive the encoded video data to be decoded via a link 16. The link 16 may comprise any type of communication medium or device capable of moving the encoded video data from the source device 12 to the destination device 14. In one example, the link 16 may comprise a communication medium to enable the source device 12 to transmit the encoded video data directly to the destination device 14 in real time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to the destination device 14. The communication medium may comprise any wireless or wired communication medium, such as a Radio Frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from the source device 12 to the destination device 14.

[0064] In some other implementations, the encoded video data may be transmitted from an output interface 22 to a storage device 32. Subsequently, the encoded video data in the storage device 32 may be accessed by the destination device 14 via an input interface 28. The storage device 32 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, Digital Versatile Disks (DVDs), Compact Disc Read-Only Memories (CD-ROMs), flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing the encoded video data. In a further example, the storage

device 32 may correspond to a file server or another intermediate storage device that may hold the encoded video data generated by the source device 12. The destination device 14 may access the stored video data from the storage device 32 via streaming or downloading. The file server may be any type of computer capable of storing the encoded video data and transmitting the encoded video data to the destination device 14. Exemplary file servers include a web server (e.g., for a website), a File Transfer Protocol (FTP) server, Network Attached Storage (NAS) devices, or a local disk drive. The destination device 14 may access the encoded video data through any standard data connection, including a wireless channel (e.g., a Wireless Fidelity (Wi-Fi) connection), a wired connection (e.g., Digital Subscriber Line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on a file server. The transmission of the encoded video data from the storage device 32 may be a streaming transmission, a download transmission, or a combination of both.

[0065] As shown in Figure 1, the source device 12 includes a video source 18, a video encoder 20 and the output interface 22. The video source 18 may include a source such as a video capturing device, e.g., a video camera, a video archive containing previously captured video, a video feeding interface to receive video from a video content provider, and/or a computer graphics system for generating computer graphics data as the source video, or a combination of such sources. As one example, if the video source 18 is a video camera of a security surveillance system, the source device 12 and the destination device 14 may form camera phones or video phones. However, the implementations described in the present application may be applicable to video coding in general, and may be applied to wireless and/or wired applications.

[0066] The captured, pre-captured, or computer-generated video may be encoded by the video encoder 20. The encoded video data may be transmitted directly to the destination device 14 via the output interface 22 of the source device 12. The encoded video data may also (or alternatively) be stored onto the storage device 32 for later access by the destination device 14 or other devices, for decoding and/or playback. The output interface 22 may further include a modem and/or a transmitter.

[0067] The destination device 14 includes the input interface 28, a video decoder 30, and a display device 34. The input interface 28 may include a receiver and/or a modem and receive the encoded video data over the link 16. The encoded video data communicated over the link 16, or provided on the storage device 32, may include a variety of syntax elements generated by the video encoder 20 for use by the video decoder 30 in decoding the video data. Such syntax elements may be included within the encoded video data transmitted on a communication medium, stored on a storage medium, or stored on a file server.

[0068] In some implementations, the destination device 14 may include the display device 34, which can be an integrated display device and an external display device that is configured to communicate with the destination device 14. The display device 34 displays the decoded

video data to a user, and may comprise any of 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.

[0069] The video encoder 20 and the video decoder 30 may operate according to proprietary or industry standards, such as VVC, HEVC, MPEG-4, Part 10, AVC, or extensions of such standards. It should be understood that the present application is not limited to a specific video encoding/decoding standard and may be applicable to other video encoding/decoding standards. It is generally contemplated that the video encoder 20 of the source device 12 may be configured to encode video data according to any of these current or future standards. Similarly, it is also generally contemplated that the video decoder 30 of the destination device 14 may be configured to decode video data according to any of these current or future standards.

[0070] The video encoder 20 and the video decoder 30 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When implemented partially in software, an electronic device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the video encoding/decoding operations disclosed in the present disclosure. Each of the video encoder 20 and the 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.

[0071] In some implementations, at least a part of components of the source device 12 (for example, the video source 18, the video encoder 20 or components included in the video encoder 20 as described below with reference to Figure 2, and the output interface 22) and/or at least a part of components of the destination device 14 (for example, the input interface 28, the video decoder 30 or components included in the video decoder 30 as described below with reference to Figure 3, and the display device 34) may operate in a cloud computing service network which may provide software, platforms, and/or infrastructure, such as Software as a Service (SaaS), Platform as a Service (PaaS), or Infrastructure as a Service (IaaS). In some implementations, one or more components in the source device 12 and/or the destination device 14 which are not included in the cloud computing service network may be provided in one or more client devices, and the one or more client devices may communicate with server computers in the cloud computing service network through a wireless communication network (for example, a cellular communication network, a short-range wireless communication network, or a global navigation satellite system (GNSS) communication network) or a wired communication network (e.g., a local area network (LAN) communication network or a power

line communication (PLC) network). In an embodiment, at least a part of operations described herein may be implemented as cloud-based services provided by one or more server computers which are implemented by the at least a part of the components of the source device 12 and/or the at least a part of the components of the destination device 14 in the cloud computing service network; and one or more other operations described herein may be implemented by the one or more client devices. In some implementations, the cloud computing service network may be a private cloud, a public cloud, or a hybrid cloud. The terms such as “cloud,” “cloud computing,” “cloud-based” etc. herein may be used interchangeably as appropriate without departing from the scope of the present disclosure. It should be understood that the present disclosure is not limited to being implemented in the cloud computing service network described above. Instead, the present disclosure may also be implemented in any other type of computing environments currently known or developed in the future.

[0072] Figure 2 is a block diagram illustrating an exemplary video encoder 20 in accordance with some implementations described in the present application. The video encoder 20 may perform intra and inter predictive coding of video blocks within video frames. Intra predictive coding relies on spatial prediction to reduce or remove spatial redundancy in video data within a given video frame or picture. Inter predictive coding relies on temporal prediction to reduce or remove temporal redundancy in video data within adjacent video frames or pictures of a video sequence. It should be noted that the term “frame” may be used as synonyms for the term “image” or “picture” in the field of video coding.

[0073] As shown in Figure 2, the video encoder 20 includes a video data memory 40, a prediction processing unit 41, a Decoded Picture Buffer (DPB) 64, a summer 50, a transform processing unit 52, a quantization unit 54, and an entropy encoding unit 56. The prediction processing unit 41 further includes a motion estimation unit 42, a motion compensation unit 44, a partition unit 45, an intra prediction processing unit 46, and an intra Block Copy (BC) unit 48. In some implementations, the video encoder 20 also includes an inverse quantization unit 58, an inverse transform processing unit 60, and a summer 62 for video block reconstruction. An in-loop filter 63, such as a deblocking filter, may be positioned between the summer 62 and the DPB 64 to filter block boundaries to remove blockiness artifacts from reconstructed video. Another in-loop filter, such as Sample Adaptive Offset (SAO) filter, Cross Component Sample Adaptive Offset (CCSAO) filter and/or Adaptive in-Loop Filter (ALF), may also be used in addition to the deblocking filter to filter an output of the summer 62. It should be illustrated that for the CCSAO technique, the present application is not limited to the embodiments described herein, and instead, the application may be applied to a situation where an offset is selected for any of a luma component, a Cb chroma component and a Cr chroma component according to any other of the luma component, the Cb chroma component and the Cr chroma component to modify said any component based on the selected offset. Further, it

should also be illustrated that a first component mentioned herein may be any of the luma component, the Cb chroma component and the Cr chroma component, a second component mentioned herein may be any other of the luma component, the Cb chroma component and the Cr chroma component, and a third component mentioned herein may be a remaining one of the luma component, the Cb chroma component and the Cr chroma component. In some examples, the in-loop filters may be omitted, and the decoded video block may be directly provided by the summer 62 to the DPB 64. The video encoder 20 may take the form of a fixed or programmable hardware unit or may be divided among one or more of the illustrated fixed or programmable hardware units.

[0074] The video data memory 40 may store video data to be encoded by the components of the video encoder 20. The video data in the video data memory 40 may be obtained, for example, from the video source 18 as shown in Figure 1. The DPB 64 is a buffer that stores reference video data (for example, reference frames or pictures) for use in encoding video data by the video encoder 20 (e.g., in intra or inter predictive coding modes). The video data memory 40 and the DPB 64 may be formed by any of a variety of memory devices. In various examples, the video data memory 40 may be on-chip with other components of the video encoder 20, or off-chip relative to those components.

[0075] As shown in Figure 2, after receiving the video data, the partition unit 45 within the prediction processing unit 41 partitions the video data into video blocks. This partitioning may also include partitioning a video frame into slices, tiles (for example, sets of video blocks), or other larger Coding Units (CUs) according to predefined splitting structures such as a Quad-Tree (QT) structure associated with the video data. The video frame is or may be regarded as a two-dimensional array or matrix of samples with sample values. A sample in the array may also be referred to as a pixel or a pel. A number of samples in horizontal and vertical directions (or axes) of the array or picture define a size and/or a resolution of the video frame. The video frame may be divided into multiple video blocks by, for example, using QT partitioning. The video block again is or may be regarded as a two-dimensional array or matrix of samples with sample values, although of smaller dimension than the video frame. A number of samples in horizontal and vertical directions (or axes) of the video block define a size of the video block. The video block may further be partitioned into one or more block partitions or sub-blocks (which may form again blocks) by, for example, iteratively using QT partitioning, Binary-Tree (BT) partitioning or Triple-Tree (TT) partitioning or any combination thereof. It should be noted that the term “block” or “video block” as used herein may be a portion, in particular a rectangular (square or non-square) portion, of a frame or a picture. With reference, for example, to HEVC and VVC, the block or video block may be or correspond to a Coding Tree Unit (CTU), a CU, a Prediction Unit (PU) or a Transform Unit (TU) and/or may be or correspond to a corresponding block, e.g. a Coding Tree Block (CTB), a Coding Block (CB), a Prediction Block (PB) or a Transform Block (TB) and/or to a sub-block.

[0076] The prediction processing unit 41 may select one of a plurality of possible predictive coding modes, such as one of a plurality of intra predictive coding modes or one of a plurality of inter predictive coding modes, for the current video block based on error results (e.g., coding rate and the level of distortion). The prediction processing unit 41 may provide the resulting intra or inter prediction coded block to the summer 50 to generate a residual block and to the summer 62 to reconstruct the encoded block for use as part of a reference frame subsequently. The prediction processing unit 41 also provides syntax elements, such as motion vectors, intra-mode indicators, partition information, and other such syntax information, to the entropy encoding unit 56.

[0077] In order to select an appropriate intra predictive coding mode for the current video block, the intra prediction processing unit 46 within the prediction processing unit 41 may perform intra predictive coding of the current video block relative to one or more neighbor blocks in the same frame as the current block to be coded to provide spatial prediction. The motion estimation unit 42 and the motion compensation unit 44 within the prediction processing unit 41 perform inter predictive coding of the current video block relative to one or more predictive blocks in one or more reference frames to provide temporal prediction. The video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[0078] In some implementations, the motion estimation unit 42 determines the inter prediction mode for a current video frame by generating a motion vector, which indicates the displacement of a video block within the current video frame relative to a predictive block within a reference video frame, according to a predetermined pattern within a sequence of video frames. Motion estimation, performed by the motion estimation unit 42, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a video block within a current video frame or picture relative to a predictive block within a reference frame relative to the current block being coded within the current frame. The predetermined pattern may designate video frames in the sequence as P frames or B frames. The intra BC unit 48 may determine vectors, e.g., block vectors, for intra BC coding in a manner similar to the determination of motion vectors by the motion estimation unit 42 for inter prediction, or may utilize the motion estimation unit 42 to determine the block vector.

[0079] A predictive block for the video block may be or may correspond to a block or a reference block of a reference frame that is deemed as closely matching the video block to be coded in terms of pixel difference, which may be determined by Sum of Absolute Difference (SAD), Sum of Square Difference (SSD), or other difference metrics. In some implementations, the video encoder 20 may calculate values for sub-integer pixel positions of reference frames stored in the DPB 64. For example, the video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel

positions of the reference frame. Therefore, the motion estimation unit 42 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

[0080] The motion estimation unit 42 calculates a motion vector for a video block in an inter prediction coded frame by comparing the position of the video block to the position of a predictive block of a reference frame selected from a first reference frame list (List 0) or a second reference frame list (List 1), each of which identifies one or more reference frames stored in the DPB 64. The motion estimation unit 42 sends the calculated motion vector to the motion compensation unit 44 and then to the entropy encoding unit 56.

[0081] Motion compensation, performed by the motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by the motion estimation unit 42. Upon receiving the motion vector for the current video block, the motion compensation unit 44 may locate a predictive block to which the motion vector points in one of the reference frame lists, retrieve the predictive block from the DPB 64, and forward the predictive block to the summer 50. The summer 50 then forms a residual video block of pixel difference values by subtracting pixel values of the predictive block provided by the motion compensation unit 44 from the pixel values of the current video block being coded. The pixel difference values forming the residual video block may include luma or chroma component differences or both. The motion compensation unit 44 may also generate syntax elements associated with the video blocks of a video frame for use by the video decoder 30 in decoding the video blocks of the video frame. The syntax elements may include, for example, syntax elements defining the motion vector used to identify the predictive block, any flags indicating the prediction mode, or any other syntax information described herein. Note that the motion estimation unit 42 and the motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes.

[0082] In some implementations, the intra BC unit 48 may generate vectors and fetch predictive blocks in a manner similar to that described above in connection with the motion estimation unit 42 and the motion compensation unit 44, but with the predictive blocks being in the same frame as the current block being coded and with the vectors being referred to as block vectors as opposed to motion vectors. In particular, the intra BC unit 48 may determine an intra-prediction mode to use to encode a current block. In some examples, the intra BC unit 48 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and test their performance through rate-distortion analysis. Next, the intra BC unit 48 may select, among the various tested intra-prediction modes, an appropriate intra-prediction mode to use and generate an intra-mode indicator accordingly. For example, the intra BC unit 48 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes as the appropriate intra-prediction mode to

use. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to produce the encoded block, as well as a bitrate (i.e., a number of bits) used to produce the encoded block. Intra BC unit 48 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

[0083] In other examples, the intra BC unit 48 may use the motion estimation unit 42 and the motion compensation unit 44, in whole or in part, to perform such functions for Intra BC prediction according to the implementations described herein. In either case, for Intra block copy, a predictive block may be a block that is deemed as closely matching the block to be coded, in terms of pixel difference, which may be determined by SAD, SSD, or other difference metrics, and identification of the predictive block may include calculation of values for sub-integer pixel positions.

[0084] Whether the predictive block is from the same frame according to intra prediction, or a different frame according to inter prediction, the video encoder 20 may form a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. The pixel difference values forming the residual video block may include both luma and chroma component differences.

[0085] The intra prediction processing unit 46 may intra-predict a current video block, as an alternative to the inter-prediction performed by the motion estimation unit 42 and the motion compensation unit 44, or the intra block copy prediction performed by the intra BC unit 48, as described above. In particular, the intra prediction processing unit 46 may determine an intra prediction mode to use to encode a current block. To do so, the intra prediction processing unit 46 may encode a current block using various intra prediction modes, e.g., during separate encoding passes, and the intra prediction processing unit 46 (or a mode selection unit, in some examples) may select an appropriate intra prediction mode to use from the tested intra prediction modes. The intra prediction processing unit 46 may provide information indicative of the selected intra-prediction mode for the block to the entropy encoding unit 56. The entropy encoding unit 56 may encode the information indicating the selected intra-prediction mode in the bitstream.

[0086] After the prediction processing unit 41 determines the predictive block for the current video block via either inter prediction or intra prediction, the summer 50 forms a residual video block by subtracting the predictive block from the current video block. The residual video data in the residual block may be included in one or more TUs and is provided to the transform processing unit 52. The transform processing unit 52 transforms the residual video data into residual transform coefficients using a transform, such as a Discrete Cosine Transform (DCT) or a conceptually similar transform.

[0087] The transform processing unit 52 may send the resulting transform coefficients to the quantization unit 54. The quantization unit 54 quantizes the transform coefficients to further

reduce the bit rate. The quantization process may also reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, the quantization unit 54 may then perform a scan of a matrix including the quantized transform coefficients. Alternatively, the entropy encoding unit 56 may perform the scan.

[0088] Following quantization, the entropy encoding unit 56 entropy encodes the quantized transform coefficients into a video bitstream using, e.g., Context Adaptive Variable Length Coding (CAVLC), Context Adaptive Binary Arithmetic Coding (CABAC), Syntax-based context-adaptive Binary Arithmetic Coding (SBAC), Probability Interval Partitioning Entropy (PIPE) coding or another entropy encoding methodology or technique. The encoded bitstream may then be transmitted to the video decoder 30 as shown in Figure 1, or archived in the storage device 32 as shown in Figure 1 for later transmission to or retrieval by the video decoder 30. The entropy encoding unit 56 may also entropy encode the motion vectors and the other syntax elements for the current video frame being coded.

[0089] The inverse quantization unit 58 and the inverse transform processing unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual video block in the pixel domain for generating a reference block for prediction of other video blocks. As noted above, the motion compensation unit 44 may generate a motion compensated predictive block from one or more reference blocks of the frames stored in the DPB 64. The motion compensation unit 44 may also apply one or more interpolation filters to the predictive block to calculate sub-integer pixel values for use in motion estimation.

[0090] The summer 62 adds the reconstructed residual block to the motion compensated predictive block produced by the motion compensation unit 44 to produce a reference block for storage in the DPB 64. The reference block may then be used by the intra BC unit 48, the motion estimation unit 42 and the motion compensation unit 44 as a predictive block to inter predict another video block in a subsequent video frame.

[0091] Figure 3 is a block diagram illustrating an exemplary video decoder 30 in accordance with some implementations of the present application. The video decoder 30 includes a video data memory 79, an entropy decoding unit 80, a prediction processing unit 81, an inverse quantization unit 86, an inverse transform processing unit 88, a summer 90, and a DPB 92. The prediction processing unit 81 further includes a motion compensation unit 82, an intra prediction unit 84, and an intra BC unit 85. The video decoder 30 may perform a decoding process generally reciprocal to the encoding process described above with respect to the video encoder 20 in connection with Figure 2. For example, the motion compensation unit 82 may generate prediction data based on motion vectors received from the entropy decoding unit 80, while the intra-prediction unit 84 may generate prediction data based on intra-prediction mode indicators received from the entropy decoding unit 80.

[0092] In some examples, a unit of the video decoder 30 may be tasked to perform the

implementations of the present application. Also, in some examples, the implementations of the present disclosure may be divided among one or more of the units of the video decoder 30. For example, the intra BC unit 85 may perform the implementations of the present application, alone, or in combination with other units of the video decoder 30, such as the motion compensation unit 82, the intra prediction unit 84, and the entropy decoding unit 80. In some examples, the video decoder 30 may not include the intra BC unit 85 and the functionality of intra BC unit 85 may be performed by other components of the prediction processing unit 81, such as the motion compensation unit 82.

[0093] The video data memory 79 may store video data, such as an encoded video bitstream, to be decoded by the other components of the video decoder 30. The video data stored in the video data memory 79 may be obtained, for example, from the storage device 32, 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 (e.g., a flash drive or hard disk). The video data memory 79 may include a Coded Picture Buffer (CPB) that stores encoded video data from an encoded video bitstream. The DPB 92 of the video decoder 30 stores reference video data for use in decoding video data by the video decoder 30 (e.g., in intra or inter predictive coding modes). The video data memory 79 and the DPB 92 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including Synchronous DRAM (SDRAM), Magneto-resistive RAM (MRAM), Resistive RAM (RRAM), or other types of memory devices. For illustrative purpose, the video data memory 79 and the DPB 92 are depicted as two distinct components of the video decoder 30 in Figure 3. But it will be apparent to one skilled in the art that the video data memory 79 and the DPB 92 may be provided by the same memory device or separate memory devices. In some examples, the video data memory 79 may be on-chip with other components of the video decoder 30, or off-chip relative to those components.

[0094] During the decoding process, the video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video frame and associated syntax elements. The video decoder 30 may receive the syntax elements at the video frame level and/or the video block level. The entropy decoding unit 80 of the video decoder 30 entropy decodes the bitstream to generate quantized coefficients, motion vectors or intra-prediction mode indicators, and other syntax elements. The entropy decoding unit 80 then forwards the motion vectors or intra-prediction mode indicators and other syntax elements to the prediction processing unit 81.

[0095] When the video frame is coded as an intra predictive coded (I) frame or for intra coded predictive blocks in other types of frames, the intra prediction unit 84 of the prediction processing unit 81 may generate prediction data for a video block of the current video frame based on a signaled intra prediction mode and reference data from previously decoded blocks of the current frame.

[0096] When the video frame is coded as an inter-predictive coded (i.e., B or P) frame, the motion compensation unit 82 of the prediction processing unit 81 produces one or more predictive blocks for a video block of the current video frame based on the motion vectors and other syntax elements received from the entropy decoding unit 80. Each of the predictive blocks may be produced from a reference frame within one of the reference frame lists. The video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference frames stored in the DPB 92.

[0097] In some examples, when the video block is coded according to the intra BC mode described herein, the intra BC unit 85 of the prediction processing unit 81 produces predictive blocks for the current video block based on block vectors and other syntax elements received from the entropy decoding unit 80. The predictive blocks may be within a reconstructed region of the same picture as the current video block defined by the video encoder 20.

[0098] The motion compensation unit 82 and/or the intra BC unit 85 determines prediction information for a video block of the current video frame by parsing the motion vectors and other syntax elements, and then uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, the motion compensation unit 82 uses some of the received syntax elements to determine a prediction mode (e.g., intra or inter prediction) used to code video blocks of the video frame, an inter prediction frame type (e.g., B or P), construction information for one or more of the reference frame lists for the frame, motion vectors for each inter predictive encoded video block of the frame, inter prediction status for each inter predictive coded video block of the frame, and other information to decode the video blocks in the current video frame.

[0099] Similarly, the intra BC unit 85 may use some of the received syntax elements, e.g., a flag, to determine that the current video block was predicted using the intra BC mode, construction information of which video blocks of the frame are within the reconstructed region and should be stored in the DPB 92, block vectors for each intra BC predicted video block of the frame, intra BC prediction status for each intra BC predicted video block of the frame, and other information to decode the video blocks in the current video frame.

[00100] The motion compensation unit 82 may also perform interpolation using the interpolation filters as used by the video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, the motion compensation unit 82 may determine the interpolation filters used by the video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[00101] The inverse quantization unit 86 inverse quantizes the quantized transform coefficients provided in the bitstream and entropy decoded by the entropy decoding unit 80 using the same quantization parameter calculated by the video encoder 20 for each video block in the video frame to determine a degree of quantization. The inverse transform processing unit 88 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a

conceptually similar inverse transform process, to the transform coefficients in order to reconstruct the residual blocks in the pixel domain.

[00102] After the motion compensation unit 82 or the intra BC unit 85 generates the predictive block for the current video block based on the vectors and other syntax elements, the summer 90 reconstructs decoded video block for the current video block by summing the residual block from the inverse transform processing unit 88 and a corresponding predictive block generated by the motion compensation unit 82 and the intra BC unit 85. An in-loop filter 91 such as deblocking filter, SAO filter, CCSAO filter and/or ALF may be positioned between the summer 90 and the DPB 92 to further process the decoded video block. In some examples, the in-loop filter 91 may be omitted, and the decoded video block may be directly provided by the summer 90 to the DPB 92. The decoded video blocks in a given frame are then stored in the DPB 92, which stores reference frames used for subsequent motion compensation of next video blocks. The DPB 92, or a memory device separate from the DPB 92, may also store decoded video for later presentation on a display device, such as the display device 34 of Figure 1.

[00103] In a typical video coding process, a video sequence typically includes an ordered set of frames or pictures. Each frame may include three sample arrays, denoted SL, SCb, and SCr. SL is a two-dimensional array of luma samples. SCb is a two-dimensional array of Cb chroma samples. SCr is a two-dimensional array of Cr chroma samples. In other instances, a frame may be monochrome and therefore includes only one two-dimensional array of luma samples.

[00104] As shown in Figure 4A, the video encoder 20 (or more specifically the partition unit 45) generates an encoded representation of a frame by first partitioning the frame into a set of CTUs. A video frame may include an integer number of CTUs ordered consecutively in a raster scan order from left to right and from top to bottom. Each CTU is a largest logical coding unit and the width and height of the CTU are signaled by the video encoder 20 in a sequence parameter set, such that all the CTUs in a video sequence have the same size being one of 128×128 , 64×64 , 32×32 , and 16×16 . But it should be noted that the present application is not necessarily limited to a particular size. As shown in Figure 4B, each CTU may comprise one CTB of luma samples, two corresponding coding tree blocks of chroma samples, and syntax elements used to code the samples of the coding tree blocks. The syntax elements describe properties of different types of units of a coded block of pixels and how the video sequence can be reconstructed at the video decoder 30, including inter or intra prediction, intra prediction mode, motion vectors, and other parameters. In monochrome pictures or pictures having three separate color planes, a CTU may comprise a single coding tree block and syntax elements used to code the samples of the coding tree block. A coding tree block may be an $N \times N$ block of samples.

[00105] To achieve a better performance, the video encoder 20 may recursively perform tree

partitioning such as binary-tree partitioning, ternary-tree partitioning, quad-tree partitioning or a combination thereof on the coding tree blocks of the CTU and divide the CTU into smaller CUs. As depicted in Figure 4C, the 64x64 CTU 400 is first divided into four smaller CUs, each having a block size of 32x32. Among the four smaller CUs, CU 410 and CU 420 are each divided into four CUs of 16x16 by block size. The two 16x16 CUs 430 and 440 are each further divided into four CUs of 8x8 by block size. Figure 4D depicts a quad-tree data structure illustrating the end result of the partition process of the CTU 400 as depicted in Figure 4C, each leaf node of the quad-tree corresponding to one CU of a respective size ranging from 32x32 to 8x8. Like the CTU depicted in Figure 4B, each CU may comprise a CB of luma samples and two corresponding coding blocks of chroma samples of a frame of the same size, and syntax elements 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. It should be noted that the quad-tree partitioning depicted in FIGS. 4C and 4D is only for illustrative purposes and one CTU can be split into CUs to adapt to varying local characteristics based on quad/ternary/binary-tree partitions. In the multi-type tree structure, one CTU is partitioned by a quad-tree structure and each quad-tree leaf CU can be further partitioned by a binary and ternary tree structure. As shown in Figure 4E, there are five possible partitioning types of a coding block having a width W and a height H, i.e., quaternary partitioning, horizontal binary partitioning, vertical binary partitioning, horizontal ternary partitioning, and vertical ternary partitioning.

[00106] In some implementations, the video encoder 20 may further partition a coding block of a CU into one or more MxN PBs. A PB is a rectangular (square or non-square) block of samples on which the same prediction, inter or intra, is applied. A PU of a CU may comprise a PB of luma samples, two corresponding PBs of chroma samples, and syntax elements used to predict the PBs. In monochrome pictures or pictures having three separate color planes, a PU may comprise a single PB and syntax structures used to predict the PB. The video encoder 20 may generate predictive luma, Cb, and Cr blocks for luma, Cb, and Cr PBs of each PU of the CU.

[00107] The video encoder 20 may use intra prediction or inter prediction to generate the predictive blocks for a PU. If the video encoder 20 uses intra prediction to generate the predictive blocks of a PU, the video encoder 20 may generate the predictive blocks of the PU based on decoded samples of the frame associated with the PU. If the video encoder 20 uses inter prediction to generate the predictive blocks of a PU, the video encoder 20 may generate the predictive blocks of the PU based on decoded samples of one or more frames other than the frame associated with the PU.

[00108] After the video encoder 20 generates predictive luma, Cb, and Cr blocks for one or more PUs of a CU, the video encoder 20 may generate a luma residual block for the CU by subtracting the CU's predictive luma blocks from its original luma coding block such that 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. Similarly, the video encoder 20 may generate a Cb residual block and a Cr residual block for the CU, respectively, such that each sample in the CU's Cb residual block indicates 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 and 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.

[00109] Furthermore, as illustrated in Figure 4C, the video encoder 20 may use quad-tree partitioning to decompose the luma, Cb, and Cr residual blocks of a CU into one or more luma, Cb, and Cr transform blocks respectively. A transform block is a rectangular (square or non-square) block of samples on which the same transform is applied. A TU of a CU may comprise a transform block of luma samples, two corresponding transform blocks of chroma samples, and syntax elements used to transform the transform block samples. Thus, each TU of a CU may be associated with a luma transform block, a Cb transform block, and a Cr transform block. In some examples, 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.

[00110] The 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. A coefficient block may be a two-dimensional array of transform coefficients. A transform coefficient may be a scalar quantity. The 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. The 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.

[00111] After generating a coefficient block (e.g., a luma coefficient block, a Cb coefficient block or a Cr coefficient block), the 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 the video encoder 20 quantizes a coefficient block, the video encoder 20 may entropy encode syntax elements indicating the quantized transform coefficients. For example, the video encoder 20 may perform CABAC on the syntax elements indicating the quantized transform coefficients. Finally, the video encoder 20 may output a bitstream that includes a sequence of bits that forms a representation of coded frames and associated data, which is either saved in the storage device 32 or transmitted to the destination device 14.

[00112] After receiving a bitstream generated by the video encoder 20, the video decoder 30 may parse the bitstream to obtain syntax elements from the bitstream. The video decoder 30 may reconstruct the frames of the video data based at least in part on the syntax elements obtained from the bitstream. The process of reconstructing the video data is generally reciprocal to the encoding process performed by the video encoder 20. For example, the video decoder 30 may perform inverse transforms on the coefficient blocks associated with TUs of a current CU to reconstruct residual blocks associated with the TUs of the current CU. The video decoder 30 also reconstructs the coding blocks of the current CU by adding the samples of the predictive blocks for PUs of the current CU to corresponding samples of the transform blocks of the TUs of the current CU. After reconstructing the coding blocks for each CU of a frame, video decoder 30 may reconstruct the frame.

[00113] As noted above, video coding achieves video compression using primarily two modes, i.e., intra-frame prediction (or intra-prediction) and inter-frame prediction (or inter-prediction). It is noted that IBC could be regarded as either intra-frame prediction or a third mode. Between the two modes, inter-frame prediction contributes more to the coding efficiency than intra-frame prediction because of the use of motion vectors for predicting a current video block from a reference video block.

[00114] But with the ever improving video data capturing technology and more refined video block size for preserving details in the video data, the amount of data required for representing motion vectors for a current frame also increases substantially. One way of overcoming this challenge is to benefit from the fact that not only a group of neighboring CUs in both the spatial and temporal domains have similar video data for predicting purpose but the motion vectors between these neighboring CUs are also similar. Therefore, it is possible to use the motion information of spatially neighboring CUs and/or temporally co-located CUs as an approximation of the motion information (e.g., motion vector) of a current CU by exploring their spatial and temporal correlation, which is also referred to as “Motion Vector Predictor (MVP)” of the current CU.

[00115] Instead of encoding, into the video bitstream, an actual motion vector of the current CU determined by the motion estimation unit 42 as described above in connection with Figure 2, the motion vector predictor of the current CU is subtracted from the actual motion vector of the current CU to produce a Motion Vector Difference (MVD) for the current CU. By doing so, there is no need to encode the motion vector determined by the motion estimation unit 42 for each CU of a frame into the video bitstream and the amount of data used for representing motion information in the video bitstream can be significantly decreased.

[00116] Like the process of choosing a predictive block in a reference frame during inter-frame prediction of a code block, a set of rules need to be adopted by both the video encoder 20 and the video decoder 30 for constructing a motion vector candidate list (also known as a “merge list”) for a current CU using those potential candidate motion vectors associated with

spatially neighboring CUs and/or temporally co-located CUs of the current CU and then selecting one member from the motion vector candidate list as a motion vector predictor for the current CU. By doing so, there is no need to transmit the motion vector candidate list itself from the video encoder 20 to the video decoder 30 and an index of the selected motion vector predictor within the motion vector candidate list is sufficient for the video encoder 20 and the video decoder 30 to use the same motion vector predictor within the motion vector candidate list for encoding and decoding the current CU.

[00117] In general, the basic inter prediction scheme applied in VVC is almost kept the same as that of HEVC, except that several prediction tools are further extended, added and/or improved, e.g., extended merge prediction, MMVD, and GPM.

[00118] Extended merge prediction

[00119] With the ever improving video data capturing technology and more refined video block size for preserving details in the video data, an amount of data required for representing motion vectors for a current picture also increases substantially. One way of overcoming this challenge is to use motion information (e.g., a motion vector) of a spatially neighboring CU, a temporally collocated CU etc. of a current CU as an approximation (e.g., prediction) of motion information of the current CU, which is also referred to as “Motion Vector Predictor (MVP)” of the current CU. The “motion vectors” used throughout the present disclosure comprise not only the motion vectors between CUs from different frames (e.g., between temporally collocated CUs in inter prediction) but also the block vectors between CUs in the same frame (e.g., between spatially neighboring CUs in intra prediction).

[00120] Like a process of choosing a predictive block in a reference picture during inter-prediction of a coding block, a set of rules need to be adopted by both the video encoder 20 and the video decoder 30 for constructing an MVP candidate list for a current CU and then selecting one MVP candidate from the MVP candidate list as an MVP for the current CU. By doing so, there is no need to transmit the MVP candidate list itself between the video encoder 20 and the video decoder 30, and an index of the MVP candidate selected from the MVP candidate list is sufficient for the video encoder 20 and the video decoder 30 to use the same MVP candidate selected from the MVP candidate list for encoding and decoding the current CU.

[00121] In VVC, the MVP candidate list is constructed by including the following five types of MVPs in order:

[00122] —Spatial MVP from spatially neighboring CUs (i.e., spatial candidates);

[00123] —Temporal MVP from temporally collocated CUs (i.e., temporal candidates);

[00124] —History-based MVP (HMVP) from a First-In-First-Out (FIFO) table;

[00125] —Pairwise average MVP; and

[00126] —Zero MVPs.

[00127] A size of the MVP candidate list is signalled in a sequence parameter set header and

a maximum allowed size of the MVP candidate list is 6. For each CU coded in merge mode, an index of the best MVP candidate is encoded using truncated unary binarization. A first bin of the index is coded with contexts and bypass coding is used for other bins of the index.

[00128] A derivation process of each type of MVPs is provided as follows. As in HEVC, VVC also supports parallel derivation of MVP candidate lists for all CUs within a certain size of area.

[00129] Derivation of MVPs from spatial candidates

[00130] The derivation of MVPs from spatial candidates (for example, CUs neighboring a current CU 101 in Figure 5) in VVC is the same as that in HEVC except that positions of first two spatial candidates are swapped. A maximum of four spatial candidates are selected from spatial candidates located at positions depicted in Figure 5, that is, a top position B0, a left position A0, a top-right position B1, a bottom-left position A1 and a top-left position B2. The derivation is performed in an order of CUs at the positions B0, A0, B1, A1 and B2. A CU at the position B2 is considered only when one or more CUs at the positions B0, A0, B1 and A1 are not available (for example, because said one or more CUs belong to other slices or tiles) or is intra coded.

[00131] After a CU at the position B0 is added as a candidate to a merge candidate list, the addition of the remaining candidates to the merge candidate list is subject to redundancy check, which ensures that candidates with the same motion information are excluded from the merge candidate list, so that coding efficiency is improved. To reduce computational complexity, not all possible candidate pairs are considered in the redundancy check. Instead, only pairs linked using a line with an arrow in Figure 6 are considered and a candidate is added to the merge candidate list only if a candidate in a corresponding pair used for the redundancy check has not the same motion information as that of the candidate to be added. Spatial MVPs derived from the candidates in the merge candidate list are added to the MVP candidate list.

[00132] Derivation of MVPs from temporal candidates

[00133] During the derivation of MVPs from temporal candidates, only one temporal candidate is added to the merge candidate list. Particularly, in the derivation of an MVP from this temporal candidate, a scaled motion vector is derived based on a collocated CU (for example, col_CU 301 in Figure 7) as the temporal candidate belonging to a collocated picture (for example, col_pic 302 in Figure 7) for a current CU (for example, curr_CU 303 in Figure 7), and is added as a temporal MVP candidate to the MVP candidate list. A reference picture list and a reference picture index to be used for derivation of the collocated CU are explicitly signalled in a slice header. The scaled motion vector is obtained (i.e., scaled) from a motion vector of the collocated CU using Picture Order Count (POC) distances, i.e., tb and td, as illustrated in Figure 7, where tb is defined to be a POC difference between a reference picture (for example, curr_ref 305 in Figure 7) of the current picture (for example, curr_pic 304 in Figure 7) and the current picture and td is defined to be a POC difference between a reference

picture (for example, col_ref 306 in Figure 7) of the collocated picture and the collocated picture. A reference picture index of the temporal candidate is set equal to zero.

[00134] A position for the temporal candidate (i.e., the collocated CU) in the current CU 401 is selected between positions C_0 and C_1 , as depicted in Figure 8. If a CU at position C_0 in the collocated picture is not available, is intra coded, or is outside of a current row of CTUs, a CU at position C_1 is used as the collocated CU for the derivation of the temporal MVP candidate. Otherwise, a CU at position C_0 is used as the collocated CU for the derivation of the temporal MVP candidate.

[00135] Derivation of HMVP candidates

[00136] HMVP candidates are added to the MVP candidate list after the spatial MVPs and the temporal MVP. Motion information of a previously coded block is stored in an HMVP table and used as an MVP for the current CU. The table with multiple HMVP candidates is maintained during the encoding/decoding process. The table is reset (emptied) when a new row of CTUs is encountered. Whenever there is a non-subblock inter-coded CU, associated motion information is added to a last entry of the HMVP table as a new HMVP candidate.

[00137] A size of the HMVP table is set to 6. When a new HMVP candidate is inserted into the HMVP table, a constrained FIFO rule is utilized, wherein redundancy check is firstly applied to find whether there is an identical HMVP in the HMVP table. If found, the identical HMVP is removed from the HMVP table and all the HMVP candidates afterwards are moved forward, and the identical HMVP is added to the last entry of the HMVP table.

[00138] HMVP candidates may be used in the MVP candidate list construction process. The latest several HMVP candidates in the HMVP table are checked in order and inserted into the MVP candidate list after the temporal MVP candidate. Redundancy check is applied on the HMVP candidates relative to the spatial candidates and/or temporal MVP candidate.

[00139] To reduce a number of redundancy check operations, the following simplifications are introduced:

[00140] —Last two entries in the HMVP table are redundancy checked relative to spatial MVP candidates derived from the spatial candidates at the positions A1 and B1, respectively; and

[00141] —Once a total number of available MVP candidates reaches the maximum allowed size of the MVP candidate list minus 1, the MVP candidate list construction process from HMVP candidates is terminated.

[00142] Derivation of pairwise average MVP candidates

[00143] Pairwise average MVP candidates are generated by averaging MVPs derived using a predefined pair of first two merge candidates in the existing merge candidate list. A first merge candidate in the predefined pair may be defined as p0Cand and a second merge candidate in the predefined pair may be defined as p1Cand. Averaged motion vectors are calculated according to availability of motion vectors of p0Cand and p1Cand separately for each reference

picture list. If both motion vectors are available for one reference picture list, these two motion vectors are averaged even when they point to different reference pictures, and a reference picture of the averaged motion vector is set to a reference picture of p0Cand; if only one motion vector is available for one reference picture list, the motion vector is used directly; if no motion vector is available for one reference picture list, the motion vector and the reference picture index for this reference picture list are kept invalid.

[00144] Zero MVPs

[00145] When the MVP candidate list is not full after the pairwise average MVP candidates are added, zero MVPs are inserted at the end of the MVP candidate list until the maximum allowed size of the MVP candidate list is reached.

[00146] MMVD

[00147] As described above, in the merge mode, motion information (i.e., an MVP candidate) is implicitly derived from an MVP candidate list constructed for a current CU and is directly used as an MV of the current CU for generation of prediction samples of the current CU, which may result in a certain error between an actual MV of the current CU and the implicitly derived MVP. In order to increase the accuracy of an MV of the current CU, MMVD is introduced in VVC where a Motion Vector Difference (MVD) of the current CU is added to the implicitly derived MVP to obtain the MV of the current CU. An MMVD flag is signalled after a regular merge flag is transmitted to specify whether an MMVD mode is used for the current CU.

[00148] In the MMVD mode, after an MVP candidate is selected from first two MVP candidates in the MVP candidate list, MMVD information is signalled, wherein the MMVD information includes an MMVD candidate flag which is used to specify which one of the first two MVP candidates is selected to be used as an MV basis, a distance index for indication of motion magnitude information of the MVD, and a direction index for indication of motion direction information of the MVD.

[00149] The distance index, which specifies the motion magnitude information of the MVD, indicates a pre-defined offset from a starting point (represented by, for example, a dotted circle in Figure 9) in a reference picture (for example, L0 reference picture 501 or L1 reference picture 503 in Figure 9) of the current CU to which the selected MVP candidate points, and the MVD may be derived from the offset and may be added to the selected MVP candidate. A relation between distance indexes and pre-defined offsets is specified in Table 1 below.

Distance index	0	1	2	3	4	5	6	7
Offset (in unit of luma samples)	1/4	1/2	1	2	4	8	16	32

Table 1

[00150] The direction index specifies a sign of the MVD, which represents a direction of the MVD relative to the starting point. Table 2 specifies a relation between direction indexes and pre-defined signs. It should be illustrated that the meaning of a sign of the MVD may be variant according to information of the selected MVP candidate. When the selected MVP candidate is an un-prediction MV or bi-prediction MVs with both MVs pointing to the same side of the current picture (i.e., POCs of two reference pictures (for example, reference pictures of list 0 and list 1, which are also referred to as L0 reference picture and L1 reference picture respectively) of the current picture are both greater than a POC of the current picture, or are both less than the POC of the current picture), the sign in Table 2 specifies the sign of the MVD added to the selected MVP candidate. When the selected MVP candidate is bi-prediction MVs with both MVs pointing to different sides of the current picture (i.e. a POC of one reference picture of the current picture is greater than the POC of the current picture, and a POC of the other reference picture of the current picture is less than the POC of the current picture), if a POC distance for L0 reference picture (i.e., a POC distance between the L0 reference picture and the current picture) is greater than a POC distance for L1 reference picture (i.e., a POC distance between the L1 reference picture and the current picture), the sign in Table 2 specifies a sign of an MVD for list 0 MVD0 added to an MVP for list 0 MVP0 of the selected MVP candidate and a sign of an MVD for list 1 MVD1 added to an MVP for list 1 MVP1 of the selected MVP candidate is opposite to the sign in Table 2; otherwise, if the POC distance for L1 reference picture is greater than the POC distance for L0 reference picture, the sign in Table 2 specifies the sign of MVD1 added to MVP1 and the sign of MVD0 added to MVP0 is opposite to the sign in Table 2.

Direction indexes	00	01	10	11
x-axis	+	−	N/A	N/A
y-axis	N/A	N/A	+	−

Table 2

[00151] The MVD is scaled according to the POC distances. If the POC distances for both L0 reference picture and L1 reference picture are the same, no scaling is needed for the MVD. Otherwise, if the POC distance for L0 reference picture is greater than the POC distance for L1 reference picture, MVD1 is scaled. If the POC distance for L1 reference picture is greater than the POC distance for L0 reference picture, MVD0 is scaled.

[00152] GPM

[00153] In VVC, GPM is supported for inter prediction. The GPM is signalled using a CU-level flag as one kind of merge mode, with other merge modes including the regular merge mode, the MMVD mode, the CIIP mode and the subblock merge mode. A total of 64 partitions are supported by GPM for each possible CU size $W \times H$ ($W = 2^m$ and $H = 2^n$, with

$m, n \in \{3, 4, 5, 6\}$) excluding 8×64 and 64×8 .

[00154] When the GPM is used, a CU is split into two parts by a geometrically located straight line. The position of the splitting line is mathematically derived from angle and offset parameters of a specific partition. Each part of the CU obtained by the geometrical partitioning is inter-predicted using its own motion; and only uni-prediction is allowed for each partition, that is, each part has one motion vector and one reference index. The uni-prediction motion constraint is applied to ensure that like the conventional bi-prediction, only two motion compensated predictions are needed for each CU.

[00155] If the GPM is used for the current CU, then a geometric partition index indicating a partition mode of the geometric partitioning (indicating an angle and an offset of the geometric partitioning), and two merge indexes (one for each partition) are further signalled.

[00156] An uni-prediction candidate list is derived directly from a merge candidate list constructed according to the extended merge prediction process described above. Denote n as an index of a uni-prediction motion vector in the uni-prediction candidate list. An LX motion vector of an n^{th} merge candidate in the merge candidate list, with X equal to a parity of n , is used as the n^{th} uni-prediction motion vector for the GPM. These motion vectors are marked with “x” in Figure 10. In a case that a corresponding LX motion vector of the n^{th} merge candidate in the merge candidate list does not exist, an $L(1 - X)$ motion vector of the same merge candidate is used instead as the uni-prediction motion vector for the GPM.

[00157] CIIP

[00158] In VVC, when a CU is coded in a merge mode, if the CU contains at least 64 luma samples (that is, a width of CU times a height of the CU is equal to or larger than 64), and if both the width and the height of the CU are less than 128 luma samples, an additional flag is signalled to indicate if a CIIP mode is applied to the current CU. In the CIIP mode, a prediction signal is obtained by combining an inter prediction signal with an intra prediction signal. The inter prediction signal in the CIIP mode is derived using the same inter prediction process as that applied in the regular merge mode; and the intra prediction signal in the CIIP mode is derived following the regular intra prediction process with a planar mode. Then, the intra prediction signal and the inter prediction signal are combined using weighted averaging, where a weight value is calculated depending on coding modes of top and left neighboring blocks of the current CU 1601 (as shown in Figure 11) as follows:

[00159] —If the top neighboring block is available and is intra coded, then isIntraTop is set to 1, otherwise isIntraTop is set to 0;

[00160] —If the left neighboring block is available and is intra coded, then isIntraLeft is set to 1, otherwise isIntraLeft is set to 0;

[00161] —If $(\text{isIntraLeft} + \text{isIntraTop})$ is equal to 2, then the weight value is set to 3;

[00162] —Otherwise, if $(\text{isIntraLeft} + \text{isIntraTop})$ is equal to 1, then the weight value is set to 2;

[00163] —Otherwise, the weight value is set to 1.

[00164] —The prediction signal P_{CIIP} in the CIIP mode is derived as follows:

$$P_{CIIP} = ((4 - wt) * P_{inter} + wt * P_{intra} + 2) \gg 2 \quad (1)$$

[00165] Where P_{inter} is the inter prediction signal in the CIIP mode, P_{intra} is the intra prediction signal in the CIIP mode, wt is the weight value, and $\gg$ represents a right shift operation.

[00166] Intra block copy in Versatile Video Coding (VVC)

[00167] Intra block copy (IBC) is a tool adopted in HEVC extensions on SCC. It is well known that it significantly improves the coding efficiency of screen content materials. Since IBC mode is implemented as a block level coding mode, block matching (BM) is performed at the encoder to find the optimal block vector (or motion vector) for each CU. Here, a block vector is used to indicate the displacement from the current block to a reference block, which is already reconstructed inside the current picture. The luma block vector of an IBC-coded CU is in integer precision. The chroma block vector rounds to integer precision as well. When combined with AMVR, the IBC mode can switch between 1-pel and 4-pel motion vector precisions. An IBC-coded CU is treated as the third prediction mode other than intra or inter prediction modes. The IBC mode is applicable to the CUs with both width and height smaller than or equal to 64 luma samples.

[00168] At the encoder side, hash-based motion estimation is performed for IBC. The encoder performs RD check for blocks with either width or height no larger than 16 luma samples. For non-merge mode, the block vector search is performed using hash-based search first. If hash search does not return valid candidate, block matching based local search will be performed.

[00169] In the hash-based search, hash key matching (32-bit CRC) between the current block and a reference block is extended to all allowed block sizes. The hash key calculation for every position in the current picture is based on 4x4 subblocks. For the current block of a larger size, a hash key is determined to match that of the reference block when all the hash keys of all 4x4 subblocks match the hash keys in the corresponding reference locations. If hash keys of multiple reference blocks are found to match that of the current block, the block vector costs of each matched reference are calculated and the one with the minimum cost is selected.

[00170] In block matching search, the search range is set to cover both the previous and current CTUs.

[00171] At CU level, IBC mode is signalled with a flag and it can be signaled as IBC AMVP mode or IBC skip/merge mode as follows:

[00172] IBC skip/merge mode: a merge candidate index is used to indicate which of the block vectors in the list from neighboring candidate IBC coded blocks is used to predict the current block. The merge list includes spatial, HMVP, and pairwise candidates.

[00173] IBC AMVP mode: block vector difference is coded in the same way as a motion

vector difference. The block vector prediction method uses two candidates as predictors, one from left neighbor and one from above neighbor (if IBC coded). When either neighbor is not available, a default block vector will be used as a predictor. A flag is signaled to indicate the block vector predictor index.

[00174] IBC reference region

[00175] To reduce memory consumption and decoder complexity, the IBC in VVC allows only the reconstructed portion of the predefined area including the region of current CTU and some region of the left CTU. Figure 12 illustrates the reference region of IBC Mode, where each block represents 64x64 luma sample unit.

[00176] Depending on the location of the current coding CU location within the current CTU, the following applies:

[00177] If current block falls into the top-left 64x64 block of the current CTU, then in addition to the already reconstructed samples in the current CTU, it can also refer to the reference samples in the bottom-right 64x64 blocks of the left CTU, using CPR mode. The current block can also refer to the reference samples in the bottom-left 64x64 block of the left CTU and the reference samples in the top-right 64x64 block of the left CTU, using CPR mode.

[00178] If current block falls into the top-right 64x64 block of the current CTU, then in addition to the already reconstructed samples in the current CTU, if luma location (0, 64) relative to the current CTU has not yet been reconstructed, the current block can also refer to the reference samples in the bottom-left 64x64 block and bottom-right 64x64 block of the left CTU, using CPR mode; otherwise, the current block can also refer to reference samples in bottom-right 64x64 block of the left CTU.

[00179] If current block falls into the bottom-left 64x64 block of the current CTU, then in addition to the already reconstructed samples in the current CTU, if luma location (64, 0) relative to the current CTU has not yet been reconstructed, the current block can also refer to the reference samples in the top-right 64x64 block and bottom-right 64x64 block of the left CTU, using CPR mode. Otherwise, the current block can also refer to the reference samples in the bottom-right 64x64 block of the left CTU, using CPR mode.

[00180] If current block falls into the bottom-right 64x64 block of the current CTU, it can only refer to the already reconstructed samples in the current CTU, using CPR mode.

[00181] This restriction allows the IBC mode to be implemented using local on-chip memory for hardware implementations.

[00182] IBC interaction with other coding tools

[00183] The interaction between IBC mode and other inter coding tools in VVC, such as pairwise merge candidate, history based motion vector predictor (HMVP), combined intra/inter prediction mode (CIIP), merge mode with motion vector difference (MMVD), and geometric partitioning mode (GPM) are as follows:

[00184] IBC can be used with pairwise merge candidate and HMVP. A new pairwise IBC

merge candidate can be generated by averaging two IBC merge candidates. For HMVP, IBC motion is inserted into history buffer for future referencing.

[00185] IBC cannot be used in combination with the following inter tools: affine motion, CIIP, MMVD, and GPM.

[00186] IBC is not allowed for the chroma coding blocks when DUAL_TREE partition is used.

[00187] Unlike in the HEVC screen content coding extension, the current picture is no longer included as one of the reference pictures in the reference picture list 0 for IBC prediction. The derivation process of motion vectors for IBC mode excludes all neighboring blocks in inter mode and vice versa. The following IBC design aspects are applied:

[00188] IBC shares the same process as in regular MV merge including with pairwise merge candidate and history-based motion predictor, but disallows TMVP and zero vector because they are invalid for IBC mode.

[00189] Separate HMVP buffer (5 candidates each) is used for conventional MV and IBC.

[00190] Block vector constraints are implemented in the form of bitstream conformance constraint, the encoder needs to ensure that no invalid vectors are present in the bitstream, and merge shall not be used if the merge candidate is invalid (out of range or 0). Such bitstream conformance constraint is expressed in terms of a virtual buffer as described below.

[00191] For deblocking, IBC is handled as inter mode.

[00192] If the current block is coded using IBC prediction mode, AMVR does not use quarter-pel; instead, AMVR is signaled to only indicate whether MV is inter-pel or 4 integer-pel.

[00193] The number of IBC merge candidates can be signalled in the slice header separately from the numbers of regular, subblock, and geometric merge candidates.

[00194] A virtual buffer concept is used to describe the allowable reference region for IBC prediction mode and valid block vectors. Denote CTU size as $ctbSize$, the virtual buffer, $ibcBuf$, has width being $wIbcBuf = 128 \times 128 / ctbSize$ and height $hIbcBuf = ctbSize$. For example, for a CTU size of 128×128 , the size of $ibcBuf$ is also 128×128 ; for a CTU size of 64×64 , the size of $ibcBuf$ is 256×64 ; and a CTU size of 32×32 , the size of $ibcBuf$ is 512×32 .

[00195] The size of a VPDU is $\min(ctbSize, 64)$ in each dimension, $W_v = \min(ctbSize, 64)$.

[00196] The virtual IBC buffer, $ibcBuf$ is maintained as follows.

[00197] At the beginning of decoding each CTU row, refresh the whole $ibcBuf$ with an invalid value -1 .

[00198] At the beginning of decoding a VPDU ($xVPDU$, $yVPDU$) relative to the top-left corner of the picture, set the $ibcBuf[x][y] = -1$, with $x = xVPDU \% wIbcBuf, \dots, xVPDU \% wIbcBuf + W_v - 1$; $y = yVPDU \% ctbSize, \dots, yVPDU \% ctbSize + W_v - 1$.

[00199] After decoding a CU contains (x, y) relative to the top-left corner of the picture, set

[00200] $ibcBuf[x \% wIbcBuf][y \% ctbSize] = recSample[x][y]$

[00201] For a block covering the coordinates (x, y), if the following is true for a block vector $bv = (bv[0], bv[1])$, then it is valid; otherwise, it is not valid:

[00202] $ibcBuf[(x + bv[0]) \% wIbcBuf][(y + bv[1]) \% ctbSize]$ shall not be equal to -1.

[00203] **Intra block copy in Enhanced Compression Model (ECM)**

[00204] In ECM, IBC is improved from below aspects.

[00205] **IBC merge/AMVP list construction**

[00206] The IBC merge/AMVP list construction is modified as follows:

[00207] Only if an IBC merge/AMVP candidate is valid, it can be inserted into the IBC merge/AMVP candidate list.

[00208] Above-right, bottom-left, and above-left spatial candidates and one pairwise average candidate can be added into the IBC merge/AMVP candidate list.

[00209] Template based adaptive reordering (ARMC-TM) is applied to IBC merge list.

[00210] The HMVP table size for IBC is increased to 25. After up to 20 IBC merge candidates are derived with full pruning, they are reordered together. After reordering, the first 6 candidates with the lowest template matching costs are selected as the final candidates in the IBC merge list.

[00211] The zero vectors' candidates to pad the IBC Merge/AMVP list are replaced with a set of BVP candidates located in the IBC reference region. A zero vector is invalid as a block vector in IBC merge mode, and consequently, it is discarded as BVP in the IBC candidate list.

[00212] Three candidates are located on the nearest corners of the reference region, and three additional candidates are determined in the middle of the three sub-regions (A, B, and C), whose coordinates are determined by the width, and height of the current block and the ΔX and ΔY parameters, as is depicted in Figure 13.

[00213] **IBC with Template Matching**

[00214] Template Matching is used in IBC for both IBC merge mode and IBC AMVP mode.

[00215] The IBC-TM merge list is modified compared to the one used by regular IBC merge mode such that the candidates are selected according to a pruning method with a motion distance between the candidates as in the regular TM merge mode. The ending zero motion fulfillment is replaced by motion vectors to the left (-W, 0), top (0, -H) and top-left (-W, -H), where W is the width and H the height of the current CU.

[00216] In the IBC-TM merge mode, the selected candidates are refined with the Template Matching method prior to the RDO or decoding process. The IBC-TM merge mode has been put in competition with the regular IBC merge mode and a TM-merge flag is signaled.

[00217] In the IBC-TM AMVP mode, up to 3 candidates are selected from the IBC-TM merge list. Each of those 3 selected candidates are refined using the Template Matching method and sorted according to their resulting Template Matching cost. Only the 2 first ones are then considered in the motion estimation process as usual.

[00218] The Template Matching refinement for both IBC-TM merge and AMVP modes is

quite simple since IBC motion vectors are constrained (i) to be integer and (ii) within a reference region as shown in Figure 12. So, in IBC-TM merge mode, all refinements are performed at integer precision, and in IBC-TM AMVP mode, they are performed either at integer or 4-pel precision depending on the AMVR value. Such a refinement accesses only to samples without interpolation. In both cases, the refined motion vectors and the used template in each refinement step must respect the constraint of the reference region.

[00219] IBC reference area

[00220] The reference area for IBC is extended to two CTU rows above. Figure 14 illustrates the reference area for coding CTU (m,n). Specifically, for CTU (m,n) to be coded, the reference area includes CTUs with index (m-2,n-2)...(W,n-2),(0,n-1)...(W,n-1),(0,n)...(m,n), where W denotes the maximum horizontal index within the current tile, slice or picture. This setting ensures that for CTU size being 128, IBC does not require extra memory in the current ETM platform. The per-sample block vector search (or called local search) range is limited to $[-(C \ll 1), C \gg 2]$ horizontally and $[-C, C \gg 2]$ vertically to adapt to the reference area extension, where C denotes the CTU size.

[00221] IBC merge mode with block vector differences

[00222] IBC merge mode with block vector differences is adopted in ECM. The distance set is {1-pel, 2-pel, 4-pel, 8-pel, 12-pel, 16-pel, 24-pel, 32-pel, 40-pel, 48-pel, 56-pel, 64-pel, 72-pel, 80-pel, 88-pel, 96-pel, 104-pel, 112-pel, 120-pel, 128-pel}, and the BVD directions are two horizontal and two vertical directions.

[00223] The base candidates are selected from the first five candidates in the reordered IBC merge list. And based on the SAD cost between the template (one row above and one column left to the current block) and its reference for each refinement position, all the possible MBVD refinement positions (20×4) for each base candidate are reordered. Finally, the top 8 refinement positions with the lowest template SAD costs are kept as available positions, consequently for MBVD index coding.

[00224] IBC adaptation for camera-captured content

[00225] When adapt IBC for camera-captured content, IBC reference range is reduced from 2 CTU rows to 2x128 rows as shown in Figure 15. At encoder side to reduce the complexity, the local search range is set to $[-8,8]$ horizontally and $[-8,8]$ vertically centered at the first block vector predictor of the current CU. This encoder modification is not applied to SCC sequences.

[00226] Combination of CIIP with TIMD and TM merge

[00227] In CIIP mode, the prediction samples are generated by weighting an inter prediction signal predicted using CIIP-TM merge candidate and an intra prediction signal predicted using TIMD derived intra prediction mode. The method is only applied to coding blocks with an area less than or equal to 1024.

[00228] The TIMD derivation method is used to derive the intra prediction mode in CIIP.

Specifically, the intra prediction mode with the smallest SATD values in the TIMD mode list is selected and mapped to one of the 67 regular intra prediction modes.

[00229] In addition, it is also proposed to modify the weights (wIntra, wInter) for the two tests if the derived intra prediction mode is an angular mode. For near-horizontal modes ($2 \leq \text{angular mode index} < 34$), the current block is vertically divided as shown in Figure 16A; for near-vertical modes ($34 \leq \text{angular mode index} \leq 66$), the current block is horizontally divided as shown in Figure 16B.

[00230] The (wIntra, wInter) for different sub-blocks are shown in Table 3.

The sub-block index	(wIntra, wInter)
0	(6,2)
1	(5,3)
2	(3,5)
3	(2,6)

Table 3. The modified weights used for angular modes.

[00231] With CIIP-TM, a CIIP-TM merge candidate list is built for the CIIP-TM mode. The merge candidates are refined by template matching. The CIIP-TM merge candidates are also reordered by the ARMC method as regular merge candidates. The maximum number of CIIP-TM merge candidates is equal to two.

[00232] Multi-hypothesis prediction (MHP)

[00233] In the multi-hypothesis inter prediction mode, one or more additional motion-compensated prediction signals are signaled, in addition to the conventional bi prediction signal. The resulting overall prediction signal is obtained by sample-wise weighted superposition. With the bi prediction signal p_{bi} and the first additional inter prediction signal/hypothesis h_3 , the resulting prediction signal p_3 is obtained as follows:

$$p_3 = (1 - \alpha)p_{bi} + \alpha h_3 \quad (2)$$

[00234] The weighting factor α is specified by the new syntax element `add_hyp_weight_idx`, according to the mapping presented in Table 4:

<code>add_hyp_weight_idx</code>	α
0	1/4
1	-1/8

Table 4. The mapping between `add_hyp_weight_idx` and α .

[00235] Analogously to above, more than one additional prediction signal can be used. The resulting overall prediction signal is accumulated iteratively with each additional prediction signal.

$$p_{n+1} = (1 - \alpha_{n+1})p_n + \alpha_{n+1}h_{n+1} \quad (3)$$

[00236] The resulting overall prediction signal is obtained as the last p_n (i.e., the p_n having the largest index n). Within this mode, up to two additional prediction signals can be used (i.e., n is limited to 2).

[00237] The motion parameters of each additional prediction hypothesis can be signaled either explicitly by specifying the reference index, the motion vector predictor index, and the motion vector difference, or implicitly by specifying a merge index. A separate multi-hypothesis merge flag distinguishes between these two signalling modes.

[00238] For inter AMVP mode, MHP is only applied if non-equal weight in BCW is selected in bi-prediction mode.

[00239] Combination of MHP and BDOF is possible, however the BDOF is only applied to the bi-prediction signal part of the prediction signal (i.e., the ordinary first two hypotheses).

[00240] **Geometric Partitioning Mode (GPM) in ECM**

[00241] **GPM with merge motion vector differences (MMVD)**

[00242] GPM in VVC is extended by applying motion vector refinement on top of the existing GPM uni-directional MVs. A flag is first signalled for a GPM CU, to specify whether this mode is used. If the mode is used, each geometric partition of a GPM CU can further decide whether to signal MVD or not. If MVD is signalled for a geometric partition, after a GPM merge candidate is selected, the motion of the partition is further refined by the signalled MVDs information. All other procedures are kept the same as in GPM.

[00243] The MVD is signaled as a pair of distance and direction, similar as in MMVD. There are nine candidate distances ($\frac{1}{4}$ -pel, $\frac{1}{2}$ -pel, 1-pel, 2-pel, 3-pel, 4-pel, 6-pel, 8-pel, 16-pel), and eight candidate directions (four horizontal/vertical directions and four diagonal directions) involved in GPM with MMVD (GPM-MMVD). In addition, when `pic_fpel_mmvd_enabled_flag` is equal to 1, the MVD is left shifted by 2 as in MMVD.

[00244] **GPM with template matching (TM)**

[00245] Template matching is applied to GPM. When GPM mode is enabled for a CU, a CU-level flag is signaled to indicate whether TM is applied to both geometric partitions. Motion information for each geometric partition is refined using TM. When TM is chosen, a template is constructed using left, above or left and above neighboring samples according to partition angle, as shown in Table 5. The motion is then refined by minimizing the difference between the current template and the template in the reference picture using the same search pattern of merge mode with half-pel interpolation filter disabled.

Partition angle	0	2	3	4	5	8	11	12	13	14
-----------------	---	---	---	---	---	---	----	----	----	----

1st partition	A	A	A	A	L+A	L+A	L+A	L+A	A	A
2nd partition	L+A	L+A	L+A	L	L	L	L	L+A	L+A	L+A
Partition angle	16	18	19	20	21	24	27	28	29	30
1st partition	A	A	A	A	L+A	L+A	L+A	L+A	A	A
2nd partition	L+A	L+A	L+A	L	L	L	L	L+A	L+A	L+A

Table 5

[00246] Table 5 shows template for the 1st and 2nd geometric partitions, where A represents using above samples, L represents using left samples, and L+A represents using both left and above samples.

[00247] A GPM candidate list is constructed as follows:

[00248] 1. Interleaved List-0 MV candidates and List-1 MV candidates are derived directly from the regular merge candidate list, where List-0 MV candidates are higher priority than List-1 MV candidates. A pruning method with an adaptive threshold based on the current CU size is applied to remove redundant MV candidates.

[00249] 2. Interleaved List-1 MV candidates and List-0 MV candidates are further derived directly from the regular merge candidate list, where List-1 MV candidates are higher priority than List-0 MV candidates. The same pruning method with the adaptive threshold is also applied to remove redundant MV candidates.

[00250] 3. Zero MV candidates are padded until the GPM candidate list is full.

[00251] The GPM-MMVD and GPM-TM are exclusively enabled to one GPM CU. This is done by firstly signaling the GPM-MMVD syntax. When both two GPM-MMVD control flags are equal to false (i.e., the GPM-MMVD are disabled for two GPM partitions), the GPM-TM flag is signaled to indicate whether the template matching is applied to the two GPM partitions. Otherwise (at least one GPM-MMVD flag is equal to true), the value of the GPM-TM flag is inferred to be false.

[00252] GPM with inter and intra prediction

[00253] In GPM with inter and intra prediction, the final prediction samples are generated by weighting inter predicted samples and intra predicted samples for each GPM-separated region. The inter predicted samples are derived by inter GPM whereas the intra predicted samples are derived by an intra prediction mode (IPM) candidate list and an index signaled from the encoder. The IPM candidate list size is pre-defined as 3. The available IPM candidates are the parallel angular mode against the GPM block boundary (Parallel mode), the perpendicular angular mode against the GPM block boundary (Perpendicular mode), and the Planar mode as shown in Figures 17A to 17C, respectively. Furthermore, GPM with intra and intra prediction as shown in Figure 17D is restricted to reduce the signalling overhead for IPMs and avoid an increase in the size of the intra prediction circuit on the hardware decoder. In

addition, a direct motion vector and IPM storage on the GPM-blending area is introduced to further improve the coding performance.

[00254] In DIMD and neighboring mode based IPM derivation Parallel mode is registered first. Therefore, max two IPM candidates derived from the decoder-side intra mode derivation (DIMD) method and/or the neighboring blocks can be registered if there is not the same IPM candidate in the list. As for the neighboring mode derivation, there are five positions for available neighboring blocks at most, but they are restricted by the angle of GPM block boundary as shown in Table 6, which are already used for GPM with template matching (GPM-TM).

Angle of GPM	0	2	3	4	5	8	11	12	13	14
1st partition	A	A	A	A	L+A	L+A	L+A	L+A	A	A
2nd partition	L+A	L+A	L+A	L	L	L	L	L+A	L+A	L+A
Partition angle	16	18	19	20	21	24	27	28	29	30
1st partition	A	A	A	A	L+A	L+A	L+A	L+A	A	A
2nd partition	L+A	L+A	L+A	L	L	L	L	L+A	L+A	L+A

Table 6

[00255] Table 6 shows the position of available neighboring blocks for IPM candidate derivation based on the angle of GPM block boundary. A and L denotes the above and left side of the prediction block.

[00256] GPM-intra can be combined with GPM with merge with motion vector difference (GPM-MMVD). TIMD is used for on IPM candidates of GPM-intra to further improve the coding performance. The Parallel mode can be registered first, then IPM candidates of TIMD, DIMD, and neighboring blocks.

[00257] Template matching based reordering for GPM split modes

[00258] In template matching based reordering for GPM split modes, given the motion information of the current GPM block, the respective TM cost values of GPM split modes are computed. Then, all GPM split modes are reordered in ascending ordering based on the TM cost values. Instead of sending GPM split mode, an index using Golomb-Rice code to indicate where the exact GPM split mode is located in the reordering list is signaled.

[00259] The reordering method for GPM split modes is a two-step process performed after the respective reference templates of the two GPM partitions in a coding unit are generated, as follows:

[00260] • extending GPM partition edge into the reference templates of the two GPM partitions, resulting in 64 reference templates and computing the respective TM cost for each of the 64 reference templates;

[00261] • reordering GPM split modes based on their TM cost values in ascending order and marking the best 32 as available split modes.

[00262] The edge on the template is extended from that of the current CU, as Figure 18 illustrates, but GPM blending process is not used in the template area across the edge.

[00263] After ascending reordering using TM cost, an index is signaled.

[00264] Intra template matching

[00265] Intra template matching prediction (Intra TMP) is a special intra prediction mode that copies the best prediction block from the reconstructed part of the current frame, whose L-shaped template matches the current template. For a predefined search range, the encoder searches for the most similar template to the current template in a reconstructed part of the current frame and uses the corresponding block as a prediction block. The encoder then signals the usage of this mode, and the same prediction operation is performed at the decoder side.

[00266] The prediction signal is generated by matching the L-shaped causal neighbor of the current block with another block in a predefined search area in Figure 19 consisting of:

[00267] R1: current CTU

[00268] R2: top-left CTU

[00269] R3: above CTU

[00270] R4: left CTU

[00271] Sum of absolute differences (SAD) is used as a cost function.

[00272] Within each region, the decoder searches for the template that has least SAD with respect to the current one and uses its corresponding block as a prediction block.

[00273] The dimensions of all regions (SearchRange_w, SearchRange_h) are set proportional to the block dimension (BlkW, BlkH) to have a fixed number of SAD comparisons per pixel. That is:

[00274] $\text{SearchRange_w} = a * \text{BlkW}$

[00275] $\text{SearchRange_h} = a * \text{BlkH}$

[00276] Where ‘ a ’ is a constant that controls the gain/complexity trade-off. In practice, ‘ a ’ is equal to 5.

[00277] The Intra template matching tool is enabled for CUs with size less than or equal to 64 in width and height. This maximum CU size for Intra template matching is configurable.

[00278] The Intra template matching prediction mode is signaled at CU level through a dedicated flag when DIMD is not used for current CU.

[00279] Fusion for template-based intra mode derivation (TIMD)

[00280] For each intra prediction mode in MPMs, The SATD between the prediction and reconstruction samples of the template is calculated. First two intra prediction modes with the minimum SATD are selected as the TIMD modes. These two TIMD modes are fused with the weights after applying PDPC process, and such weighted intra prediction is used to code the current CU. Position dependent intra prediction combination (PDPC) is included in the

derivation of the TIMD modes.

[00281] The costs of the two selected modes are compared with a threshold, in the test the cost factor of 2 is applied as follows:

[00282] $\text{costMode2} < 2 * \text{costMode1}$.

[00283] If this condition is true, the fusion is applied, otherwise the only mode1 is used.

[00284] Weights of the modes are computed from their SATD costs as follows:

[00285] $\text{weight1} = \text{costMode2} / (\text{costMode1} + \text{costMode2})$

[00286] $\text{weight2} = 1 - \text{weight1}$

[00287] The division operations are conducted using the same lookup table (LUT) based integerization scheme used by the CCLM.

[00288] **Local illumination compensation (LIC)**

[00289] LIC is an inter prediction technique to model local illumination variation between current block and its prediction block as a function of that between current block template and reference block template. The parameters of the function can be denoted by a scale α and an offset β , which forms a linear equation, that is, $\alpha * p[x] + \beta$ to compensate illumination changes, where $p[x]$ is a reference sample pointed to by MV at a location x on reference picture. When wrap around motion compensation is enabled, the MV shall be clipped with wrap around offset taken into consideration. Since α and β can be derived based on current block template and reference block template, no signaling overhead is required for them, except that an LIC flag is signaled for AMVP mode to indicate the use of LIC.

[00290] The local illumination compensation proposed in JVET-O0066 is used for uni-prediction inter CUs with the following modifications:

[00291] intra neighbor samples can be used in LIC parameter derivation;

[00292] LIC is disabled for blocks with less than 32 luma samples;

[00293] for both non-subblock and affine modes, LIC parameter derivation is performed based on the template block samples corresponding to the current CU, instead of partial template block samples corresponding to first top-left 16x16 unit; and

[00294] samples of the reference block template are generated by using MC with the block MV without rounding it to integer-pel precision.

[00295] **OBMC**

[00296] When OBMC is applied, top and left boundary pixels of a CU are refined using neighboring block's motion information with a weighted prediction as described in JVET-L0101.

[00297] Conditions of not applying OBMC are as follows:

[00298] when OBMC is disabled at SPS level;

[00299] when current block has intra mode or IBC mode;

[00300] when current block applies LIC; and

[00301] when current luma block area is smaller or equal to 32.

[00302] A subblock-boundary OBMC is performed by applying the same blending to the top, left, bottom, and right subblock boundary pixels using neighboring subblocks' motion information. It is enabled for the subblock based coding tools:

[00303] affine AMVP modes;

[00304] affine merge modes and subblock-based temporal motion vector prediction (SbTMVP); and

[00305] subblock-based bilateral matching.

[00306] When OBMC mode is used in CIIP mode with LMCS, inter blending is performed prior to LMCS mapping of inter samples. LMCS is applied to blended inter samples which are combined with LMCS applied intra samples in CIIP mode,

$$Inter'_{predY} = \frac{(128 - w_1) \times Inter_{predY} + w_1 \times OBMC_{predY}}{128}$$

$$PredY = \frac{(4 - w_0) \times FwdMap(Inter'_{predY}) + w_0 \times Intra_{predY}}{4}$$

[00307] where $Inter_{predY}$ represents the samples predicted by the motion of current block in the original domain, $Intra_{predY}$ represents the samples predicted in the mapped domain, $OBMC_{predY}$ represents the samples predicted by the motion of neighboring blocks in the original domain, and w_0 and w_1 are the weights.

[00308] **Template matching based OBMC**

[00309] In template matching based OBMC scheme, instead of directly using the weighted prediction, the prediction value of CU boundary samples derivation approach is decided according to the template matching costs, including using current block's motion information only, or using neighboring block's motion information as well with one of the blending modes.

[00310] In this scheme for each block with a size of 4×4 at the top CU boundary, the above template size equals to 4×1 . If N adjacent blocks have the same motion information, then the above template size is enlarged to $4N \times 1$ since the MC operation can be processed at one time. For each left block with a size of 4×4 at the left CU boundary, the left template size equals to 1×4 or $1 \times 4N$ (Figure 20).

[00311] For each 4×4 top block (or N 4×4 blocks group), the prediction value of boundary samples is derived following the below steps.

[00312] Take block A as the current block and its above neighboring block AboveNeighbor_A for example. The operation for left blocks is conducted in the same manner.

[00313] First, three template matching costs (Cost1, Cost2, Cost3) are measured by SAD between the reconstructed samples of a template and its corresponding reference samples derived by MC process according to the following three types of motion information:

[00314] Cost1 is calculated according to A's motion information.

- [00315] Cost2 is calculated according to AboveNeighbor_A's motion information.
- [00316] Cost3 is calculated according to weighted prediction of A's and AboveNeighbor_A's motion information with weighting factors as 3/4 and 1/4 respectively.
- [00317] Second, choose one approach to calculate the final prediction results of boundary samples by comparing Cost1, Cost2 and Cost 3.
- [00318] The original MC result using current block's motion information is denoted as Pixel1, and the MC result using neighboring block's motion information is denoted as Pixel2. The final prediction result is denoted as NewPixel.
- [00319] If Cost1 is minimum, then $\text{NewPixel}(i,j) = \text{Pixel1}(i,j)$.
- [00320] If $(\text{Cost2} + (\text{Cost2} \gg 2) + (\text{Cost2} \gg 3)) \leq \text{Cost1}$, then blending mode 1 is used.
- [00321] For luma blocks, the number of blending pixel rows is 4.
- [00322] $\text{NewPixel}(i,0) = (26 \times \text{Pixel1}(i,0) + 6 \times \text{Pixel2}(i,0) + 16) \gg 5$
- [00323] $\text{NewPixel}(i,1) = (7 \times \text{Pixel1}(i,1) + \text{Pixel2}(i,1) + 4) \gg 3$
- [00324] $\text{NewPixel}(i,2) = (15 \times \text{Pixel1}(i,2) + \text{Pixel2}(i,2) + 8) \gg 4$
- [00325] $\text{NewPixel}(i,3) = (31 \times \text{Pixel1}(i,3) + \text{Pixel2}(i,3) + 16) \gg 5$
- [00326] For chroma blocks, the number of blending pixel rows is 1.
- [00327] $\text{NewPixel}(i,0) = (26 \times \text{Pixel1}(i,0) + 6 \times \text{Pixel2}(i,0) + 16) \gg 5$
- [00328] If $\text{Cost1} \leq \text{Cost2}$, then blending mode 2 is used.
- [00329] For luma blocks, the number of blending pixel rows is 2.
- [00330] $\text{NewPixel}(i,0) = (15 \times \text{Pixel1}(i,0) + \text{Pixel2}(i,0) + 8) \gg 4$
- [00331] $\text{NewPixel}(i,1) = (31 \times \text{Pixel1}(i,1) + \text{Pixel2}(i,1) + 16) \gg 5$
- [00332] For chroma blocks, the number of blending pixel rows/columns is 1.
- [00333] $\text{NewPixel}(i,0) = (15 \times \text{Pixel1}(i,0) + \text{Pixel2}(i,0) + 8) \gg 4$
- [00334] Otherwise, blending mode 3 is used.
- [00335] For luma blocks, the number of blending pixel rows is 4.
- [00336] $\text{NewPixel}(i,1) = (7 \times \text{Pixel1}(i,1) + \text{Pixel2}(i,1) + 4) \gg 3$
- [00337] $\text{NewPixel}(i,2) = (15 \times \text{Pixel1}(i,2) + \text{Pixel2}(i,2) + 8) \gg 4$
- [00338] $\text{NewPixel}(i,3) = (31 \times \text{Pixel1}(i,3) + \text{Pixel2}(i,3) + 16) \gg 5$
- [00339] For chroma blocks, the number of blending pixel rows is 1.
- [00340] $\text{NewPixel}(i,0) = (7 \times \text{Pixel1}(i,0) + \text{Pixel2}(i,0) + 4) \gg 3$
- [00341] **Adaptive Reordering of Merge Candidates with Template Matching (ARMC-TM)**
- [00342] The reordering method is applied to regular merge mode, template matching (TM) merge mode, and affine merge mode (excluding the SbTMVP candidate). For the TM merge mode, merge candidates are reordered before the refinement process.
- [00343] After a merge candidate list is constructed, merge candidates are divided into several subgroups. The subgroup size is set to 5 for regular merge mode and TM merge mode. The subgroup size is set to 3 for affine merge mode. Merge candidates in each subgroup are

reordered ascendingly according to cost values based on template matching. For simplification, merge candidates in the last but not the first subgroup are not reordered.

[00344] The template matching cost of a merge candidate is measured by the sum of absolute differences (SAD) between samples of a template of the current block and their corresponding reference samples. The template comprises a set of reconstructed samples neighboring to the current block. Reference samples of the template are located by the motion information of the merge candidate.

[00345] When a merge candidate utilizes bi-directional prediction, the reference samples of the template of the merge candidate are also generated by bi-prediction as shown in FIG. 21.

[00346] For subblock-based merge candidates with subblock size equal to $W_{sub} \times H_{sub}$, the above template comprises several sub-templates with the size of $W_{sub} \times 1$, and the left template comprises several sub-templates with the size of $1 \times H_{sub}$. As shown in FIG. 22, the motion information of the subblocks in the first row and the first column of current block is used to derive the reference samples of each sub-template.

[00347] Direct block vector for chroma block

[00348] The direct block vector is used for chroma block in dual tree slices. When chroma dual tree is activated, a flag is signaled to indicate whether a chroma block is coded using IBC mode. If one of the luma blocks in five locations shown in FIG. 23 is coded with IBC or intraTMP mode, its block vector is scaled and is used as block vector for the chroma block. Template matching is used to perform block vector scaling.

[00349] Although the existing IBC scheme can provide significant improvement of intra coding in the ECM, there is room to further improve its performance. Meanwhile, some parts of the existing convolutional cross-component model (CCCM) mode also need to be simplified for efficient codec hardware implementations or improved for better coding efficiency. Furthermore, the trade-off between its implementation complexity and its coding efficiency benefit needs to be further improved.

[00350] In this disclosure, to address the issues as pointed out above, methods are provided to further improve the existing design of the IBC. In general, the main features of the proposed technologies in this disclosure are summarized as follows.

[00351] The IBC prediction is filtered with CCCM tool. Filtered intra block copy (FIBC) is a special intra prediction mode that applies a filter on the IBC-based prediction block to increase prediction accuracy and adapt the characteristics of the copied block to the local neighbourhood.

[00352] In FIBC, training samples may be adjacent to the current block. It is known that reference from local regions can improve the accuracy of prediction in prediction.

[00353] In FIBC, training samples may be not adjacent to the current block. It is known that reference from non-local regions can also improve the accuracy of prediction in prediction.

[00354] In FIBC, only one hypothesis may be utilized, i.e., the best matching block which

leads to the minimum matching cost is selected as the final prediction.

[00355] In FIBC, multiple hypothesis may also be utilized.

[00356] It should be understood that the figures in this disclosure may be combined with all examples mentioned in this disclosure and the disclosed methods may be applied independently or jointly.

[00357] Filtered Intra Block Copy (FIBC)

[00358] According to one or more embodiments of the disclosure, IBC prediction is filtered with CCCM tool. Different methods may be used to achieve this goal. The existing CCCM mode applies a variety of filters for predicting the chroma sample value based on corresponding luma sample values. Unlike CCCM, filtered intra block copy (FIBC) is a special intra prediction mode that applies a filter on the IBC-based prediction block to predict the target luma or chroma sample of the current block based on the corresponding luma or chroma samples of the reference block respectively, in order to increase prediction accuracy and adapt the characteristics of the copied block to the local neighbourhood.

[00359] According to one or more embodiments of the disclosure, IBC prediction is further filtered. Different methods may be used to achieve this goal. Filtered intra block copy (FIBC) is a special intra prediction mode that applies a filter on the intra block copy based prediction block to increase prediction accuracy and adapt the characteristics of the copied block to the local neighbourhood.

[00360] According to one or more embodiments of the disclosure, reconstructed luma/chroma samples over the template area of the reference block are used as inputs to the filter during training phase and corresponding reconstructed luma/chroma sample in the template area of the current block is the target. In one example, Figure 25 illustrates one filter shape (cross-shaped) and the training area for the reference block. It should be understood that for this filter shape, both the template area and the boundary region of the template area can be part of the training area for the reference block. Reconstructed samples in the boundary region may be used in training when available and they are padded with closest available sample when unavailable. On the other hand, a training area for the current block can be determined as the template area of the current block. In the filtering phase where the filter coefficients of the filter have been trained/determined through the training phase, the filter may be applied to the corresponding sample values of reference block and the boundary region of the reference block to predict each of the sample values for the current block.

[00361] According to one or more embodiments of the disclosure, the filter coefficients (i.e., parameters) are derived using the regression based MSE minimization technique (i.e., LDL decomposition) existing in ECM and being utilized by other tools such as CCCM.

[00362] According to one or more embodiments of the disclosure, the convolutional N-tap (N is an integer and larger than 1) filter may include $(N-1-M)$ -tap (M is an integer) spatial terms, M nonlinear terms and a bias term. The $(N-1-M)$ -tap spatial terms correspond to

neighboring sample values, such as luma samples (i.e., $L_0, L_1, \dots, L_8$), from the reconstructed reference block as illustrated in Figure 26. In this example, the formula for each new prediction luma sample is as follows:

$$\text{predVal} = \sum_{i=0}^{N-2-M} \alpha_i \cdot (L_i - \text{offsetLuma}) + \sum_{i=0}^{M-1} \alpha_{i+N-M-1} \cdot (((L_i - \text{offsetLuma})^2 + \beta) \gg \text{bitDepth}) + \alpha_{N-1} \cdot \beta + \text{offsetLuma}$$

[00363] Where α_i is the coefficient associated with L_i and β is the offset (i.e., $1 \ll (\text{bitDepth} - 1)$). Reference luma sample value for the top-left sample adjacent to the current block can be used as the offsetLuma value. The position and number of spatial term and nonlinear term may be different. Examples of different shape/number of filter taps as illustrated in Figure 27. For another examples, using different position and number as shown in following table.

Number of terms	Position of terms
1	Any one of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., (L_0) or (L_1) or (L_2) or (L_3) or (L_4) or (L_5) or (L_6) or (L_7) or (L_8)
2	Any two of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., (L_0, L_1) or (L_0, L_2) or (L_0, L_3) or (L_0, L_4) or (L_0, L_5) or (L_1, L_2) or (L_1, L_3) or (L_1, L_4) or (L_1, L_5) or (L_2, L_3) or (L_2, L_4) or (L_2, L_5) or (L_3, L_4) or (L_3, L_5) or (L_4, L_5)
3	Any three of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., (L_0, L_1, L_2) or (L_0, L_1, L_3) or (L_0, L_1, L_4) or (L_0, L_1, L_5) or (L_0, L_2, L_3) or (L_0, L_2, L_4) or (L_0, L_2, L_5) or (L_0, L_3, L_4) or (L_0, L_3, L_5) or (L_0, L_4, L_5) or (L_1, L_2, L_3) or (L_1, L_2, L_4) or (L_1, L_2, L_5) or (L_1, L_3, L_4) or (L_1, L_3, L_5) or (L_1, L_4, L_5) or (L_2, L_3, L_4) or (L_2, L_3, L_5) or (L_2, L_4, L_5) or (L_3, L_4, L_5)
4	Any four of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., (L_2, L_3, L_4, L_5) or (L_1, L_3, L_4, L_5) or (L_1, L_2, L_4, L_5) or (L_1, L_2, L_3, L_5) or (L_1, L_2, L_3, L_4) or (L_0, L_3, L_4, L_5) or (L_0, L_2, L_4, L_5) or (L_0, L_2, L_3, L_5) or (L_0, L_2, L_3, L_4) or (L_0, L_1, L_4, L_5) or (L_0, L_1, L_3, L_5) or (L_0, L_1, L_3, L_4) or (L_0, L_1, L_2, L_5) or (L_0, L_1, L_2, L_4) or (L_0, L_1, L_2, L_3)
5	Any five of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., $(L_0, L_1, L_2, L_3, L_4)$ or $(L_0, L_1, L_2, L_3, L_5)$ or $(L_0, L_1, L_2, L_4, L_5)$ or $(L_0, L_1, L_3, L_4, L_5)$ or $(L_0, L_2, L_3, L_4, L_5)$ or $(L_1, L_2, L_3, L_4, L_5)$
6	Any six of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., $(L_0, L_1, L_2, L_3, L_4, L_5)$
7	Any seven of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., $(L_0, L_1, L_2, L_3, L_4, L_5, L_6)$
8	Any eight of $\{L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8\}$, i.e., $(L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7)$
9	$(L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8)$

[00364] According to one or more embodiments of the disclosure, the number of filter taps may be predefined or signaled/switched in SPS/DPS/VPS/SEI/APS/PPS/PH/SH/Region/CTU/CU/Subblock/Sample levels.

[00365] According to one or more embodiments of the disclosure, template size and shapes may be same as in intra TMP, the template size used for training is 4 lines above and to the left of the current block depending on their availability.

[00366] According to one or more embodiments of the disclosure, the template size used for training is up to 5 lines above and to the left of the current block depending on their availability.

[00367] According to one or more embodiments of the disclosure, template size and shapes may be same as in CCCM, the template size used for training is 6 lines above and to the left of the current block depending on their availability.

[00368] According to one or more embodiments of the disclosure, the template size used for training may be N lines above and to the left of the current block depending on their availability, N is an integer.

[00369] According to one or more embodiments of the disclosure, the template size used for training may be N lines above of the current block depending on their availability, N is an integer.

[00370] According to one or more embodiments of the disclosure, the template size used for training may be N lines left of the current block depending on their availability, N is an integer.

[00371] According to one or more embodiments of the disclosure, the reference samples/template area of the reference block / template area of the current block may be predefined or signaled/switched in different coding levels, such as SPS/DPS/VPS/SEI/APS/PPS/PH/SH/Region/CTU/CU/Subblock/Sample levels.

[00372] According to one or more embodiments of the disclosure, location information may be used to calculate model parameters, including utilizing horizontal/vertical/diagonal distance and their non-linear term, one or more location information may be used for the purpose. In one example, the location based parameter is related to the vertical and horizontal coordinates (X_c , Y_c) of the center luma sample and it is calculated with respect to the top-left coordinates (X_{tl} , Y_{tl}) of the block, e.g. $X_c - X_{tl} + Y_c - Y_{tl}$. In another example, the location based parameters are related to the vertical and horizontal coordinates (X_c , Y_c) of the center luma sample and they are calculated with respect to the top-left coordinates (X_{tl} , Y_{tl}) of the block, e.g. $X_c - X_{tl} + Y_c - Y_{tl}$, $X_c - X_{tl}$, $Y_c - Y_{tl}$. In yet another example, the location based parameter is related to the vertical and horizontal coordinates (X_c , Y_c) of the center luma sample and it is calculated with respect to the top-left coordinates (X_{tl} , Y_{tl}) of the block, e.g. $(X_c - X_{tl} + Y_c - Y_{tl})/N$, where N is predefined number, such as 2. In yet another example, the location based parameters are related to the vertical and horizontal coordinates (X_c , Y_c) of the center luma sample and they are calculated with respect to the top-left coordinates (X_{tl} , Y_{tl}) of the block, e.g. $(X_c - X_{tl} + Y_c - Y_{tl})/N_1$, $(X_c - X_{tl})/N_2$, $(Y_c - Y_{tl})/N_3$, where $N_1 \sim N_3$ are predefined numbers, such as 2, 3 and 4. In yet another example, the location based nonlinear terms are represented as power of two of the horizontal/vertical/diagonal distance, e.g. $(X_c - X_{tl} + Y_c - Y_{tl}) * (X_c - X_{tl} + Y_c - Y_{tl})$, $(X_c - X_{tl}) * (X_c - X_{tl})$, $(Y_c - Y_{tl}) * (Y_c - Y_{tl})$, where (X_c , Y_c) are vertical and horizontal coordinates of

the center luma sample and (Xtl, Ytl) are top-left coordinates.

[00373] According to one or more embodiments of the disclosure, one enablement flag can be signaled in the bitstream to indicate the FIBC mode used. The enablement flag can be signaled in different coding levels, such as SPS/DPS/VPS/SEI/APS/PPS/PH/SH/Region/CTU/CU/Subblock/Sample levels.

[00374] According to one or more embodiments of the disclosure, instead of explicitly signaling the selected mode flag, the mode flag can be derived at decoder to save bit overhead.

[00375] According to one or more embodiments of the disclosure, no additional control flag is required and the FIBC mode will be derived under some predefined condition (e.g., specific modes, specific block sizes, specific partitions). When the predefined condition is matched, the FIBC mode will be derived based on previous decoded information.

[00376] According to one or more embodiments of the disclosure, samples in regions non-adjacent to the current block can be used to derive a model for the current block. In one embodiment, one candidate region list with N candidates may be constructed by checking potential $M \times M$ regions in order. If a checked region is available, it is put into the candidate region list. For example, a candidate region list with 6 candidates is constructed by checking potential 8×8 regions in order. The top-left positions of the potential 8×8 regions are predetermined as $\{(-xStep, 0), (0, -yStep), (xStep, -yStep), (-xStep, yStep), (-xStep, -yStep), (-2 * xStep, 0), (0, -2 * yStep), (-2 * xStep, 2 * yStep), (2 * xStep, -2 * yStep), (-2 * xStep, yStep), (xStep, -2 * yStep), (-2 * xStep, -yStep), (-xStep, -2 * yStep), (-2 * xStep, -2 * yStep), (-xStep/2, 0), (0, -yStep/2), (xStep/2, -yStep/2), (-xStep/2, yStep/2), (-xStep/2, -yStep/2)\}$, where $xStep = \text{Max}(\text{width}, 16)$, $yStep = \text{Max}(\text{height}, 16)$. Figure 28 show some possible positions of candidate regions.

[00377] According to one or more embodiments of the disclosure, one non-adjacent neighboring candidates with N candidates may be constructed by positions and inclusion order of the spatial non-adjacent neighboring candidates from two sets of spatial non-adjacent neighboring candidates in inter merge mode. If a checked region is available, it is put into the candidate region list. Figure 29 shows some possible positions of candidates.

[00378] According to one or more embodiments of the disclosure, inherited parameters of FIBC from previously decoded TB/CB/slice/picture/sequence level may be used in the current block. According to one or more embodiments of the disclosure, one control flag is signaled in TB/CB/slice/picture/sequence level to indicate whether the signaling of inherited FIBC is enabled or disabled. When the control flag is signaled as enabled, a flag of inherited FIBC is further signaled to decoder to indicate whether the inherited FIBC is used or not at signaled level.

[00379] According to one or more embodiments of the disclosure, the derived parameters of FIBC from previous decoded TB/CB/slice/picture/sequence level can be stored and used as current FIBC (which is called inherited FIBC). In one embodiment, a history-based FIBC (H-

FIBC) table may be maintained similar to the HMVP table. In one embodiment, one index value can be signaled in the bitstream to indicate which candidate model in the H-FIBC table is selected. In one embodiment, after decoding a FIBC coded block, the corresponding table may be updated. In one embodiment, the size of H-FIBC table is N. N is an integer (e.g. 4, 5, 6, 7).

[00380] Multi-hypothesis FIBC

[00381] According to one or more embodiments of the disclosure, more than one prediction block candidates are used and weighted to generate the final prediction of the current block. Assume that N prediction block candidates are used.

[00382] Prediction block candidate derivation

[00383] In one embodiment, the prediction block candidates are searched and selected according to the criterion of minimizing template matching cost, i.e., the top N candidates which lead to the minimum template matching cost are selected. The template matching cost can be not limited to SAD (sum of absolute difference) and SSE (sum of square error).

[00384] In one embodiment, the prediction block candidates may be selected according to the predefine mode, i.e., planar mode.

[00385] In one embodiment, the prediction block candidates may be selected according to the neighbor predefine mode, i.e., top predefine mode, left predefine mode.

[00386] Fixed multi-hypothesis FIBC

[00387] In this embodiment, the weighting factors to generate the final prediction block are predefined and fixed at both the encoder and decoder side. As an example, equal weighting factors can be used, i.e., $1/N$ for all the candidate blocks.

[00388] Adaptive multi-hypothesis intra FIBC

[00389] To adapt to the diverse characteristics of video content, adaptive multi-hypothesis intra FIBC methods are also proposed.

[00390] In one embodiment, the weighting factors can be derived based on the template matching costs. Denote the template matching costs of the N candidates as $C_1, C_2, \dots, C_N$, the weighting factors are calculated as follow.

$$\omega_i = \frac{1}{N-1} - \frac{C_i}{(N-1) \sum_{k=1}^N C_k}, i = 1, 2, \dots, N \quad (4)$$

[00391] It should be noted that the template matching cost can be measured with (but not limited to) SAD and SSE.

[00392] In yet another embodiment, the weighting factors can be derived/switched based on the block size or syntax element signaled in different coding levels, such as SPS/DPS/VPS/SEI/APS/PPS/PH/SH/Region/CTU/CU/Subblock/Sample levels.

[00393] In yet another embodiment, the weighting factors can be derived at the encoder side and then signaled in the bitstream to the decoder. Denote the N prediction block candidates as $P_1, P_2, \dots, P_N$ and the current block as X, then the weighting factors can be solved by the

following equation:

$$\sum_{i=1}^N \omega_i P_i = X \quad (5)$$

[00394] Equation (5) can be solved using Wiener-Hopf equations as ALF. The derived filter coefficients are then quantized to integer type and signaled in the block level.

[00395] In yet another embodiment, the weighting factors are derived based on the templates and the derived weighting factors are applied to the prediction block candidates to generate the final prediction block. Denote the templates of the prediction candidates as $T_1, T_2, \dots, T_N$ and the current block as T , then the weighting factors can be derived using the following equation:

$$\sum_{i=1}^N \omega_i T_i = T \quad (6)$$

[00396] Equation (6) can be solved using Wiener-Hopf equations. Then the final prediction block can be calculated as $\sum_{i=1}^N \omega_i P_i$, where P_i represents the i -th prediction block candidate.

[00397] FIBC mode exploits the nonlocal correlation to improve the prediction accuracy, in which similar blocks are searched and used to generate the final prediction block. In this embodiment, it is proposed to combine the nonlocal mean filtering and multi-hypothesis FIBC, which is described as follow. In the first step, N prediction block candidates are searched and identified as conducted in the FIBC. In the second step, the weighting factor is calculated as follows.

$$\omega_i = \frac{1}{Z[i]} e^{-\frac{D_i}{h^2}} \quad (7)$$

[00398] Where D_i is used to measure the distance between the template of the i -th prediction block candidate and the template of the current block, h is used as the degree of weighting and $Z[i]$ is the normalization constant:

$$Z[i] = \sum_{i=1}^N e^{-\frac{D_i}{h^2}} \quad (8)$$

[00399] To calculate the weighting factor in equation (7), the strength of weighting should be determined first. In this disclosure, several methods are proposed to decide the weighting strength.

[00400] In the first method, a weighting strength candidate list including some typical weighting strength values is defined and fixed at both encoder and decoder side. At the encoder side, the weighting strength values are checked using rate distortion optimization and the optimal weighting strength value is identified and signaled in the bitstream to the decoder side.

[00401] In the second method, the weighting strength value is estimated using the template of the prediction block candidates and the template of the current block. Denote the templates of the prediction candidates as $T_1, T_2, \dots, T_N$ and the current block as T . Then the weighting strength value can be solved using the following equation:

$$\sum_{i=1}^N \omega_i T_i = \sum_{i=1}^N \frac{1}{Z[i]} e^{-\frac{D_i}{h^2}} T_i = T \quad (9)$$

[00402] In the third method, the weighting strength value can be estimated using the QP

value and variance of the template of the current block, i.e., the relationship between the weighting strength value, QP value and the template variance can be fitted offline.

[00403] To better exploit the nonlocal correlation in the FIBC, in this embodiment singular value decomposition (SVD) is utilized to generate the final prediction block from the prediction block candidates. The width and height of the current block are denoted as W and H , the area of the current block is denoted as $d = H \times W$.

[00404] Step1. K prediction block candidates y_i are searched and identified as conducted in the FIBC.

[00405] Step2. The K prediction block candidates of the current block y construct the block group G and are arranged as a matrix:

$$Y_G = [y_G^{(1)}, y_G^{(2)}, \dots, y_G^{(K)}] \quad (10)$$

[00406] Where Y_G is a matrix with size of $d \times K$ by arranging every candidate in group G as a column vector.

[00407] Step3. Perform SVD decomposition on the matrix Y_G .

$$\text{SVD}(Y_G) = U_G \Lambda_G V_G^* \quad (11)$$

[00408] Step4. Apply soft-thresholding operation on the singular value matrix Λ_G .

$$\Lambda_{G,\tau} = \text{softTh}(\Lambda_G, \tau) \quad (12)$$

[00409] Where $\text{softTh}()$ is a function which shrinks the diagonal elements of Λ_G with the threshold τ . For the k -th diagonal element in Λ_G , it is shrunk by the nonlinear function $D_{\tau(k)}$ at level $\tau(k)$:

$$D_{\tau(k)}: \lambda_{k,\tau(k)} = \max(|\lambda_k| - \tau(k), 0) \quad (13)$$

[00410] $\Lambda_{G,\tau}$ is the matrix composed of the shrunk singular values, $\lambda_{k,\tau(k)}$ at diagonal positions.

[00411] Step5. Perform inverse SVD to obtain the filtered patch group.

$$\hat{X}_G = U_G \Lambda_{G,\tau} V_G^* \quad (14)$$

[00412] One of the key steps is to determine the thresholding values for each diagonal elements in step 4. In this invention, the thresholding values are calculated as follows. The threshold is estimated for each group of image patches with the following equation:

$$\tau_G(k) = c \times \frac{\sigma_{n,G}^2}{\sigma_{x,G,k}} \quad (15)$$

[00413] Where $\sigma_{n,G}$ is the standard deviation of noise, and $\sigma_{x,G,k}$ is the standard deviation of the original block in the k -th dimension of SVD space for group G . The deviation of the original block in SVD space is estimated as follow.

$$\sigma_{x,G,k} = \sqrt{\max\left(\frac{\lambda_{G,k}^2}{\min(d,K)} - \omega \times \sigma_{n,G}^2, 0\right)} \quad (16)$$

[00414] Where $\lambda_{G,k}^2$ is the k -th singular value of Y_{G_i} . When $\sigma_{x,G,k}$ is zero, the soft-thresholding operation is skipped. In addition, the deviation of noise is estimated with the

deviation of the predicted block using a power function which is parameterized with α and β .

$$\sigma_n = \alpha \times \sigma_y^\beta \quad (17)$$

[00415] Where σ_y is calculated as follows,

$$\sigma_y = \frac{1}{K} \sum_{k=0}^K \sqrt{\sum_{i=1}^{p^2} \frac{(y_k(i) - \mu_k)^2}{p^2}}, \quad \mu_k = \frac{1}{p^2} \sum_{i=1}^{p^2} y_k(i) \quad (18)$$

[00416] Here $y_k(i)$ represents the i -th pixel of prediction block candidate vector y_k .

[00417] **Multi-hypothesis FIBC signaling**

[00418] In this disclosure, the proposed multi-hypothesis FIBC can be utilized as a replacement of the current FIBC mode or the encoder can adaptively select FIBC mode or multi-hypothesis FIBC mode.

[00419] In one embodiment, the proposed multi-hypothesis FIBC is used as a replacement of the current FIBC mode, i.e., always using multiple hypothesis for prediction.

[00420] In yet another embodiment, one of the multi-hypothesis FIBC methods in the above sections is used jointly with the current FIBC mode. A flag is signaled in the bitstream to indicate whether multi-hypothesis FIBC mode is applied to the CU.

[00421] In yet another embodiment, more than one multi-hypothesis FIBC methods in the above sections is used jointly with the current FIBC mode. A flag is firstly signaled in the bitstream to indicate whether multi-hypothesis FIBC mode is applied. Then an index is signaled to indicate which of the multi-hypothesis FIBC methods is applied to the CU.

[00422] **Harmonization of the filter used for the FIBC mode and the FTMP mode**

[00423] The TMP prediction may also be filtered with CCCM tool, which is called the filtered template matching prediction (FTMP) mode. The process of the FTMP mode is the same as that of the FIBC mode, except that the FTMP mode does not require a signaled block vector from the encoder for finding the reference block. Instead, in the FTMP mode, the reference block may be determined at the decoder side by searching for the most similar L-shaped template to the current template in a reconstructed part of the current frame and uses the corresponding block as a reference block for the current block to be predicted. In other words, the L-shaped template associated with the reference block is the most similar template to the L-shaped template associated with the current block in the reconstructed part of the frame. This operation for determining the reference block is the same as that of the Intra TMP mode. After the reference block is determined, the same filtering process as the FIBC mode is applied for predicting the target luma or chroma sample of the current block based on the corresponding luma or chroma samples of the reference block respectively. For example, as is shown in Figure 25, the cross-shaped filter may be applied to the corresponding sample values (sample values of the reference block and the boundary region of the reference block) to predict each of the sample values for the current block.

[00424] According to one or more embodiments of the disclosure, the same filter shape

and/or template area may be applied to both the prediction under the FIBC mode and the FTMP mode. For example, before deciding which mode to apply, the decoder and/or encoder may try both the FIBC mode and the FTMP mode with the same filter shape and/or template area for better performance or reduced cost. Different methods may be used to achieve this goal.

[00425] In a first example, it is proposed to apply the filter operations that are used under the FTMP mode to the FIBC mode as well. In one example, a 6 taps filter (cross-shaped with 5 spatial components and a bias term) and the template area used for training (4 lines in width, in terms of samples, above and to the left of the current block depending on their availability) used for the FTMP mode can also be applied to the FIBC mode in a same CU.

[00426] In a second example, it is proposed to apply the filter operations that are used under the FIBC mode to the FTMP mode as well. In one example, a 2 taps filter (a single-sample filter with 1 spatial component and a bias term) and the template area used for training (1 line above and to the left of the current block depending on their availability) used for the FIBC mode can also be applied to the FTMP mode in a same CU.

[00427] Harmonization of the filter used for the FIBC mode, the FTMP mode and the CCCM mode

[00428] Filters are used in the convolutional cross-component model (CCCM) mode for predicting chroma sample values based on corresponding luma sample values. In CCCM mode, a set of chroma sample values in the reconstructed area of the current block to be predicted, along with the corresponding luma sample values for the set of chroma sample values, are used for determining the filter coefficients for the CCCM filter. In one example, the chroma sample value to be predicted and its corresponding luma sample values are collocated sample values. Although the training result for the filter coefficients may be different, the same filter shape and/or template area may be reused among CCCM, FIBC and FTMP for better performance/reduced cost. In one example, the filter shape and/or the template area may be signaled by the encoder to the decoder. In another example, the filter shape and/or the template area may be derived by the encoder based on predetermined rules, e.g., from a predefined set of candidates.

[00429] According to one or more embodiments of the disclosure, the same filter shape and/or template area may be applied to at least two of the FIBC mode, the FTMP mode and the CCCM mode. Different methods may be used to achieve this goal.

[00430] In a first example, it is proposed to apply the filter operations that are used under the CCCM mode to the FIBC mode as well. In one example, a 7 taps filter (cross-shaped with 5 spatial components, a non-linear term and a bias term) and the template area used for training (6 lines above and to the left of the current block depending on their availability) used for the CCCM mode can also be applied to the FIBC mode in a same CU.

[00431] In a second example, it is proposed to apply one of the filter operations that are used under the CCCM mode to both the FIBC mode and the FTMP mode as well. In one example,

a 11 taps filter (3*3 square with 9 spatial components, a non-linear term and a bias term) and the template area used for training (6 lines above and to the left of the current block depending on their availability) used for the CCCM mode can also be applied to the FIBC mode and the FTMP mode.

[00432] Figure 30 illustrates a workflow of a method 3000 for video decoding according to one or more aspects of the present disclosure.

[00433] At step 3010, the method 3000 comprises determining a reference block in a reconstructed part of a video frame for predicting a current block in the video frame, wherein a L-shaped template associated with the reference block is the most similar template to a L-shaped template associated with the current block in the reconstructed part of the video frame.

[00434] At step 3020, the method 3000 comprises obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a training area associated with the reference block and a training area associated with the current block.

[00435] At step 3030, the method 3000 comprises deriving, with the set of filter coefficients and the filter shape, predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block.

[00436] At step 3040, the method 3000 comprises reconstructing the current block based on the predicted sample values.

[00437] Figure 31 illustrates a workflow of a method 3100 for video encoding according to one or more aspects of the present disclosure.

[00438] At step 3110, the method 3100 comprises partitioning a video frame into a plurality of blocks.

[00439] At step 3120, the method 3100 comprises determining a reference block in a reconstructed part of the video frame for predicting a current block in the video frame, wherein a L-shaped template associated with the reference block is the most similar template to a L-shaped template associated with the current block in the reconstructed part of the video frame.

[00440] At step 3130, the method 3100 comprises obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a training area associated with the reference block and a training area associated with the current block.

[00441] At step 3140, the method 3100 comprises deriving, with the set of filter coefficients and the filter shape, predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block.

[00442] At step 3150, the method 3100 comprises generating a bitstream based on the predicted sample values.

[00443] Figure 32 illustrates a workflow of a method 3200 for video decoding according to one or more aspects of the present disclosure.

[00444] At step 3210, the method 3200 comprises determining at least one of a filter shape and a template area, wherein the at least one of the filter shape and the template area is to be

used in at least two of the filtered intra block copy (FIBC) mode, the filtered template matching prediction (FTMP) mode and the convolutional cross-component model (CCCM) mode for predicting sample values of a current block in a video frame.

[00445] At step 3220, the method 3200 comprises training, with the template area, each of sets of filter coefficients corresponding to the filter shape for the at least two of the FIBC mode, the FTMP mode and the CCCM mode respectively.

[00446] At step 3230, the method 3200 comprises deriving, with the sets of filter coefficients, predicted sample values of the current block.

[00447] At step 3240, the method 3200 comprises reconstructing the current block based on the predicted sample values.

[00448] In one example, training the set of filter coefficients for the FIBC mode comprises: determining a reference block in a reconstructed part of the video frame for predicting the current block, wherein the reference block is determined based on a block vector received from a bitstream; and obtaining the set of filter coefficients for the FIBC mode based on the sample values from both a training area associated with the reference block and a training area associated with the current block, wherein the training area associated with the reference block and the training area associated with the current block are determined based at least in part on the template area for the FIBC mode.

[00449] In one example, training the set of filter coefficients for the FTMP mode comprises: determining a reference block in a reconstructed part of the video frame for predicting the current block, wherein a L-shaped template associated with the reference block is the most similar template to a L-shaped template associated with the current block in the reconstructed part of the video frame; and obtaining the set of filter coefficients for the FTMP mode based on the sample values from both a training area associated with the reference block and a training area associated with the current block, wherein the training area associated with the reference block and the training area associated with the current block are determined based at least in part on the template area for the FTMP mode.

[00450] In one example, training the set of filter coefficients for the CCCM mode comprises: determining a set of chroma sample values in the template area; and obtaining the set of filter coefficients for the CCCM mode based on the set of chroma sample values and the corresponding luma sample values for the set of chroma sample values.

[00451] In one example, the filter shape comprises at least one of: a cross-shaped filter shape corresponding to 5 spatial terms, a non-linear term and a bias term; a single-sample filter shape corresponding to 1 spatial term and a bias term; or a 3*3 square filter shape corresponding to 9 spatial terms, a non-linear term and a bias term.

[00452] In one example, the template area comprises at least one of: 4 lines above and to the left of the current block; 1 line above and to the left of the current block; or 6 lines above and to the left of the current block.

[00453] Figure 33 illustrates a workflow of a method 3300 for video encoding according to one or more aspects of the present disclosure.

[00454] At step 3310, the method 3300 comprises partitioning a video frame into a plurality of blocks.

[00455] At step 3320, the method 3300 comprises determining at least one of a filter shape and a template area, wherein the at least one of the filter shape and the template area is to be used in at least two of the filtered intra block copy (FIBC) mode, the filtered template matching prediction (FTMP) mode and the convolutional cross-component model (CCCM) mode for predicting sample values of a current block in the video frame.

[00456] At step 3330, the method 3300 comprises training, with the template area, each of sets of filter coefficients corresponding to the filter shape for the at least two of the FIBC mode, the FTMP mode and the CCCM mode respectively.

[00457] At step 3340, the method 3300 comprises deriving, with the sets of filter coefficients, predicted sample values of the current block.

[00458] At step 3350, the method 3300 comprises generating a bitstream based on the predicted sample values.

[00459] In one example, training the set of filter coefficients for the FIBC mode comprises: determining a reference block in a reconstructed part of the video frame for predicting the current block, wherein the reference block is determined based on a block vector to be transmitted via the bitstream; and obtaining the set of filter coefficients for the FIBC mode based on the sample values from both a training area associated with the reference block and a training area associated with the current block, wherein the training area associated with the reference block and the training area associated with the current block are determined based at least in part on the template area for the FIBC mode.

[00460] In one example, training the set of filter coefficients for the FTMP mode comprises: determining a reference block in a reconstructed part of the video frame for predicting the current block, wherein a L-shaped template associated with the reference block is the most similar template to a L-shaped template associated with the current block in the reconstructed part of the video frame; and obtaining the set of filter coefficients for the FTMP mode based on the sample values from both a training area associated with the reference block and a training area associated with the current block, wherein the training area associated with the reference block and the training area associated with the current block are determined based at least in part on the template area for the FTMP mode.

[00461] In one example, training the set of filter coefficients for the CCCM mode comprises: determining a set of chroma sample values in the template area; and obtaining the set of filter coefficients for the CCCM mode based on the set of chroma sample values and the corresponding luma sample values for the set of chroma sample values.

[00462] In one example, the filter shape comprises at least one of: a cross-shaped filter shape

corresponding to 5 spatial terms, a non-linear term and a bias term; a single-sample filter shape corresponding to 1 spatial term and a bias term; or a 3*3 square filter shape corresponding to 9 spatial terms, a non-linear term and a bias term.

[00463] In one example, the template area comprises at least one of: 4 lines above and to the left of the current block; 1 line above and to the left of the current block; or 6 lines above and to the left of the current block.

[00464] Reference area and padding process in Filtered Intra Block Copy (FIBC)

[00465] According to one or more embodiments of the disclosure, the filter coefficients are calculated by minimising MSE between predicted and reconstructed luma and/or chroma samples in the reference area. In one example, FIG. 25 illustrates the reference area which consists of luma/chroma samples above and left of the CU. The extensions to the area shown in blue (slash part) are needed to support the “side samples” of the plus shaped spatial filter and different methods may be used to achieve this goal.

[00466] In the first method, it is proposed to pad with closest available sample when unavailable.

[00467] In the second method, it is proposed to pad with closest available sample no matter it is unavailable or not.

[00468] According to one or more embodiments of the disclosure, the reference area may extend one CU width to the right and one CU height below the CU boundaries.

[00469] According to one or more embodiments of the disclosure, the reference area may be adjusted to include only available samples. In one example, the reference area consists of N lines of luma/chroma samples above and left of the CU. N is an integer number and/or has a maximum upper bound (e.g. 4, 5, 6, 7).

[00470] Adaptive Reordering of Merge Candidates with Filtered Intra Block Copy (FIBC)

[00471] According to one or more embodiments of the disclosure, in process of ARMC-TM which is extended to IBC merge list, when a merge candidate utilizes FIBC prediction, the reference samples of the template of the merge candidate are also generated by FIBC.

[00472] According to one or more embodiments of the disclosure, the filter coefficients are calculated by minimising MSE between predicted and reconstructed luma and/or chroma samples in the reference area.

[00473] According to one or more embodiments of the disclosure, template size and shapes may be not included in the reference samples of the template of the merge candidate. In one example, FIG. 34 illustrates the reference area which consists of 3 lines luma samples above and left of the CU.

[00474] Merge Candidates with Filtered Intra Block Copy

[00475] According to one or more embodiments of the disclosure, IBC predictions from merge candidates are further filtered. Different methods may be used to achieve this goal.

[00476] According to one or more embodiments of the disclosure, one enable flag can be signaled in the bitstream to indicate the FIBC merge mode used. The enable flag can be signaled in SPS/DPS/VPS/SEI/APS/PPS/PH/SH/Region/CTU/CU/Subblock/Sample levels.

[00477] According to one or more embodiments of the disclosure, instead of explicitly signaling the selected mode flag, the mode flag can be derived at decoder to save bit overhead.

[00478] **Direct block vector for chroma block with Filtered Intra Block Copy**

[00479] According to one or more embodiments of the disclosure, direct block vector for chroma block are further filtered. Different methods may be used to achieve this goal.

[00480] According to one or more embodiments of the disclosure, one enable flag can be signaled in the bitstream to indicate the FIBC merge mode used. The enable flag can be signaled in SPS/DPS/VPS/SEI/APS/PPS/PH/SH/Region/CTU/CU/Subblock/Sample levels.

[00481] According to one or more embodiments of the disclosure, instead of explicitly signaling the selected mode flag, the mode flag can be inherited from luma blocks at decoder to save bit overhead.

[00482] Figure 35 illustrates a workflow of a method 3500 for video decoding according to one or more aspects of the present disclosure. The method 3500 may be performed by a decoder (e.g., video decoder 30 of Figure 3).

[00483] At step 3510, a bitstream comprising video frames may be received, and to predict a current block in a video frame, a reference block in the same video frame may be determined. For example, the reference block may be determined by using a block vector indicated in the bitstream for the current block. Here, a block vector is used to indicate the displacement from the current block to a reference block, which is already reconstructed inside the current video frame.

[00484] At step 3520, a set of filter coefficients corresponding to a filter shape may be obtained by training on the sample values of a template area associated with the reference block and the sample values of a template area associated with the current block. The template area associated with the reference block may be extended based on the filter shape, for example, to further include samples values for the edge points of the filter shape.

[00485] In one example, the template area associated with the reference block comprising samples on the left of and above the reference block may be extended to further include one or more lines on the right of the reference block and one or more lines below the reference block. For example, a line may correspond to a row or column of a picture or frame in which samples are located.

[00486] In another example, the template area associated with the reference block may be extended in four directions from the reference block, i.e., to the left and right, upward, and downward, to further include one or more additional lines from four sides around the reference block.

[00487] In one or more examples, the extended one or more lines may correspond to one

CU width or one CU height. The sample values in the extended lines may be padded with closest available sample values. In another example, the samples padded with the closest available sample values are unavailable samples, for example, the samples on the right of the reference block or below the reference block that have not been reconstructed yet.

[00488] In another example, the template area associated with the reference block may be extended to include only available samples. For example, the template area associated with the reference block is extended to include up to N lines on the left of the reference block and up to N lines above the reference block, and N is an integer (e.g., 4, 5, 6, 7).

[00489] At step 3530, each of predicted sample values of the current block may be derived based on a plurality of corresponding sample values associated with the reference block by using the set of filter coefficients and the filter shape. For example, a plurality of corresponding sample values associated with the reference block may be used as inputs to the obtained filter, to output each of the predicted sample values of the current block. The input sample values associated with the reference block may comprise a sample value in the extended template area.

[00490] At step 3540, the current block may be reconstructed based on the predicted sample values. For example, the current block may be reconstructed by combining the predicted sample values with residuals carried in the bitstream.

[00491] Figure 36 illustrates a workflow of a method 3600 for video encoding according to one or more aspects of the present disclosure. The method 3600 may be performed by an encoder (e.g., video decoder 20 of Figure 2). The steps of method 3600 may be a counterpart to that of method 3500.

[00492] At step 3610, a reference block in a video frame captured by a camera may be determined for predicting a current block in the same video frame. For example, the determination may be made based on block matching (BM) performed at the encoder.

[00493] At step 3620, a filter with a set of filter coefficients and a filter shape may be obtained by training on the sample values of a template area associated with the reference block and the sample values of a template area associated with the current block, wherein the template area associated with the reference block is extended based on the filter shape. For example, the template area associated with the reference block is extended in the same way as described with reference to the method 3500.

[00494] At step 3630, each of predicted sample values of the current block may be derived based on a plurality of corresponding sample values associated with the reference block by using the obtained filter.

[00495] At step 3640, a bitstream may be generated based on the predicted sample values.

[00496] Figure 37 illustrates a workflow of a method 3700 for video decoding according to one or more aspects of the present disclosure. The method 3700 may be performed by a decoder (e.g., video decoder 30 of Figure 3).

[00497] At step 3710, a merge candidate list for intra block copy (IBC) prediction of a current block may be obtained. For example, the merge candidate list for IBC prediction may be constructed according to the description above or other criterions. The merge candidate list may comprise a plurality of candidates that have been encoded with IBC, each having a block vector for example.

[00498] At step 3720, the plurality of candidates in the merge candidate list may be reordered based on a template matching score of each of the plurality of candidates, the template matching score is calculated based on a difference between sample values of a template of a candidate and corresponding reference sample values of a reference template of a reference block pointed by the block vector of that candidate. In the calculation of the template matching score, in response to determining that a candidate is coded with filtered IBC (i.e., FIBC), the corresponding reference sample values of the reference template are obtained also using filtered IBC (i.e., FIBC).

[00499] At step 3730, the current block may be reconstructed based on the reordered merge candidate list.

[00500] In one example, an IBC prediction of the current block may be obtained from a candidate of the merge candidate list, for example, by using the block vector of that candidate. A determination is made on whether the IBC prediction of the current block is filtered or not.

[00501] In one example, the determination is made based on a syntax element conveyed in the bitstream or an inference derived at the decoder. For example, determining that the current block is reconstructed by using filtered IBC may be made based on a syntax element or an inference without a syntax element. In response to determining that the current block is reconstructed by using filtered IBC, FIBC is performed on the IBC prediction.

[00502] In one example, at least part of the template and at least part of the reference template are not used for obtaining the template matching score. For example, as shown in figure 34 by grids, the samples intermediately close to the reference block and the candidate are not used to calculate the template matching score.

[00503] Figure 38 illustrates a workflow of a method 3800 for video encoding according to one or more aspects of the present disclosure. The method 3800 may be performed by an encoder (e.g., video decoder 20 of Figure 2). The steps of method 3800 may be a counterpart to that of method 3700.

[00504] At step 3810, a merge candidate list for intra block copy (IBC) prediction of a current block may be obtained. For example, the merge candidate list for IBC prediction may be constructed according to the description above or other criterions. The merge candidate list may comprise a plurality of candidates that have been encoded with IBC, each having a block vector for example.

[00505] At step 3820, the plurality of candidates in the merge candidate list may be reordered based on a template matching score of each of the plurality of candidates, the

template matching score is calculated based on a difference between sample values of a template of a candidate and corresponding reference sample values of a reference template of a reference block pointed by the block vector of that candidate. In the calculation of the template matching score, in response to determining that a candidate is coded with filtered IBC (i.e., FIBC), the corresponding reference sample values of the reference template are obtained also using filtered IBC (i.e., FIBC).

[00506] At step 3830, a bitstream may be generated by encoding the current block based on the reordered merge candidate list.

[00507] In one example, an IBC prediction of the current block may be obtained from a candidate of the merge candidate list, for example, by using the block vector of that candidate. A determination is made on whether the IBC prediction of the current block is filtered or not. If the determination is made that the IBC prediction of the current block is to be filtered, FIBC is performed on the IBC prediction.

[00508] In one example, a syntax element indicative of the determination that the IBC prediction of the current block is to be filtered may be conveyed in the bitstream.

[00509] Figure 39 illustrates a workflow of a method 3900 for video decoding according to one or more aspects of the present disclosure. The method 3900 may be performed by a decoder (e.g., video decoder 30 of Figure 3).

[00510] At step 3910, a block vector for a chroma block from a bitstream may be determined by using a block vector of a luma block that is associated with the chroma block, wherein the luma block has been coded with IBC. For example, the block vector for the chroma block may be directly inherited from the luma block.

[00511] At step 3920, an IBC prediction of the chroma block may be obtained based on the inherited block vector.

[00512] At step 3930, in response to determining that a filtered IBC prediction is used for the chroma block, the IBC prediction of the chroma block may be filtered to obtain the filtered IBC prediction, i.e., using FIBC.

[00513] In one example, the determination that a filtered IBC prediction is used for the chroma block may be made based on a syntax element in the bitstream.

[00514] In another example, the mode to be used by the chroma block (e.g., FIBC or IBC) may be directly inherited from the mode used by the associated luma block (e.g., FIBC or IBC), such that explicit signaling can be saved.

[00515] At step 3940, the chroma block may be reconstructed based on the filtered IBC prediction.

[00516] In one example, the filter shape and/or the filter coefficients for the chroma block may be calculated by using templates of the chroma blocks at the decoder (e.g., in a similar way as that for a luma block).

[00517] In another example, the filter shape and/or the filter coefficients for the chroma

block may be directly inherited from that of the associated luma block.

[00518] Figure 40 illustrates a workflow of a method 4000 for video encoding according to one or more aspects of the present disclosure. The method 4000 may be performed by an encoder (e.g., video decoder 20 of Figure 2). The steps of method 4000 may be a counterpart to that of method 3900.

[00519] At step 4010, a block vector for a chroma block may be determined based on a luma block that is associated with the chroma block, wherein the luma block has been coded with an intra block copy (IBC).

[00520] At step 4020, an IBC prediction of the chroma block may be obtained based on the block vector.

[00521] At step 4030, the IBC prediction of the chroma block may be filtered.

[00522] In one example, a syntax element indicating that the IBC prediction of the chroma block is to be filtered may be generated.

[00523] In another example, the FIBC mode used by the associated luma block indicating that the IBC prediction of the chroma block is to be filtered may be generated.

[00524] At step 4040, a bitstream may be generated based on the filtered IBC prediction.

[00525] In one example, the syntax element indicating that the IBC prediction of the chroma block is to be filtered may be conveyed in the bitstream.

[00526] In another example, the syntax element indicating that the IBC prediction of the chroma block is to be filtered may not be conveyed in the bitstream if it could be directly inherited from the mode used by the associated luma block.

[00527] Figure 41 shows a computing environment 4110 coupled with a user interface 4150. The computing environment 4110 can be part of a data processing server. The computing environment 4110 includes a processor 4120, a memory 4130, and an Input/Output (I/O) interface 4140.

[00528] The processor 4120 typically controls overall operations of the computing environment 4110, such as the operations associated with display, data acquisition, data communications, and image processing. The processor 4120 may include one or more processors to execute instructions to perform all or some of the steps in the above-described methods. Moreover, the processor 4120 may include one or more modules that facilitate the interaction between the processor 4120 and other components. The processor may be a Central Processing Unit (CPU), a microprocessor, a single chip machine, a Graphical Processing Unit (GPU), or the like.

[00529] The memory 4130 is configured to store various types of data to support the operation of the computing environment 4110. The memory 4130 may include predetermined software 4132. Examples of such data includes instructions for any applications or methods operated on the computing environment 4110, video datasets, image data, etc. The memory 4130 may be implemented by using any type of volatile or non-volatile memory devices, or a

combination thereof, such as a Static Random Access Memory (SRAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), an Erasable Programmable Read-Only Memory (EPROM), a Programmable Read-Only Memory (PROM), a Read-Only Memory (ROM), a magnetic memory, a flash memory, a magnetic or optical disk.

[00530] The I/O interface 4140 provides an interface between the processor 4120 and peripheral interface modules, such as a keyboard, a click wheel, buttons, and the like. The buttons may include but are not limited to, a home button, a start scan button, and a stop scan button. The I/O interface 4140 can be coupled with an encoder and decoder.

[00531] In an embodiment, there is also provided a non-transitory computer-readable storage medium comprising a plurality of programs, for example, in the memory 4130, executable by the processor 4120 in the computing environment 4110, for performing the above-described methods and/or storing a bitstream generated by the encoding method described above or a bitstream to be decoded by the decoding method described above. In one example, the plurality of programs may be executed by the processor 4120 in the computing environment 4110 to receive (for example, from the video encoder 20 in Figure 2) a bitstream or data stream including encoded video information (for example, video blocks representing encoded video frames, and/or associated one or more syntax elements, etc.), and may also be executed by the processor 4120 in the computing environment 4110 to perform the decoding method described above according to the received bitstream or data stream. In another example, the plurality of programs may be executed by the processor 4120 in the computing environment 4110 to perform the encoding method described above to encode video information (for example, video blocks representing video frames, and/or associated one or more syntax elements, etc.) into a bitstream or data stream, and may also be executed by the processor 4120 in the computing environment 4110 to transmit the bitstream or data stream (for example, to the video decoder 30 in Figure 3). Alternatively, the non-transitory computer-readable storage medium may have stored therein a bitstream or a data stream comprising encoded video information (for example, video blocks representing encoded video frames, and/or associated one or more syntax elements etc.) generated by an encoder (for example, the video encoder 20 in Figure 2) using, for example, the encoding method described above for use by a decoder (for example, the video decoder 30 in Figure 3) in decoding video data. The non-transitory computer-readable storage medium may be, for example, a ROM, a Random Access Memory (RAM), a CD-ROM, a magnetic tape, a floppy disc, an optical data storage device or the like.

[00532] In an embodiment, there is provided a bitstream generated by the encoding method described above or a bitstream to be decoded by the decoding method described above. In an embodiment, there is provided a bitstream comprising encoded video information generated by the encoding method described above or encoded video information to be decoded by the decoding method described above.

[00533] In an embodiment, there is also provided a computing device comprising one or more

processors (for example, the processor 4120); and the non-transitory computer-readable storage medium or the memory 4130 having stored therein a plurality of programs executable by the one or more processors, wherein the one or more processors, upon execution of the plurality of programs, are configured to perform the above-described methods.

[00534] In an embodiment, there is also provided a computer program product having instructions for storage or transmission of a bitstream comprising encoded video information generated by the encoding method described above or encoded video information to be decoded by the decoding method described above. In an embodiment, there is also provided a computer program product comprising a plurality of programs, for example, in the memory 4130, executable by the processor 4120 in the computing environment 4110, for performing the above-described methods. For example, the computer program product may include the non-transitory computer-readable storage medium.

[00535] In an embodiment, the computing environment 4110 may be implemented with one or more ASICs, DSPs, Digital Signal Processing Devices (DSPDs), Programmable Logic Devices (PLDs), FPGAs, GPUs, controllers, micro-controllers, microprocessors, or other electronic components, for performing the above methods.

[00536] In an embodiment, there is also provided a method of storing a bitstream, comprising storing the bitstream on a digital storage medium, wherein the bitstream comprises encoded video information generated by the encoding method described above or encoded video information to be decoded by the decoding method described above.

[00537] In an embodiment, there is also provided a method for transmitting a bitstream generated by the encoder described above. In an embodiment, there is also provided a method for receiving a bitstream to be decoded by the decoder described above.

[00538] The description of the present disclosure has been presented for purposes of illustration and is not intended to be exhaustive or limited to the present disclosure. Many modifications, variations, and alternative implementations will be apparent to those of ordinary skill in the art having the benefit of the teachings presented in the foregoing descriptions and the associated drawings.

[00539] Unless specifically stated otherwise, an order of steps of the method according to the present disclosure is only intended to be illustrative, and the steps of the method according to the present disclosure are not limited to the order specifically described above, but may be changed according to practical conditions. In addition, at least one of the steps of the method according to the present disclosure may be adjusted, combined or deleted according to practical requirements.

[00540] The examples were chosen and described in order to explain the principles of the disclosure and to enable others skilled in the art to understand the disclosure for various implementations and to best utilize the underlying principles and various implementations with various modifications as are suited to the particular use contemplated. Therefore, it is to be

understood that the scope of the disclosure is not to be limited to the specific examples of the implementations disclosed and that modifications and other implementations are intended to be included within the scope of the present disclosure.

CLAIMS

What is claimed is:

1. A method for decoding video data, comprising:
 - determining a reference block in a video frame from a bitstream for predicting a current block in the video frame;
 - obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a template area associated with the reference block and a template area associated with the current block, wherein the template area associated with the reference block is extended based on the filter shape;
 - deriving, with the set of filter coefficients and the filter shape, each of predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block; and
 - reconstructing the current block based on the predicted sample values.
2. The method of claim 1, wherein the template area associated with the reference block is extended to include one or more lines on the right of the reference block and one or more lines below the reference block.
3. The method of claim 2, wherein the template area associated with the reference block is extended to obtain an extended template area to further include one or more lines to the left of the reference block and one or more lines above the reference block.
4. The method of claim 3, wherein samples in the extended template area are padded with closest available sample values.
5. The method of claim 4, wherein the samples padded with the closest available sample values are unavailable samples.
6. The method of claim 1, wherein the template area associated with the reference block is extended to include up to N lines on the left of the reference block and up to N lines above the reference block, and N is an integer.
7. The method of claim 1, wherein the plurality of corresponding sample values

associated with the reference block comprise a sample value in the extended template area.

8. A method for decoding video data, comprising:

obtaining a merge candidate list for intra block copy (IBC) prediction of a current block, the merge candidate list comprises a plurality of candidates that have been encoded with IBC; reordering the plurality of candidates in the merge candidate list based on a template matching score of each of the plurality of candidates, the template matching score is obtained based on a difference between sample values of a template of a candidate and corresponding reference sample values of a reference template of a reference block of that candidate, wherein in response to determining that a candidate is coded with filtered IBC, the corresponding reference sample values of the reference template are obtained using filtered IBC; and reconstructing the current block based on the reordered merge candidate list.

9. The method of claim 8, further comprising:

in response to determining that the current block is reconstructed by using filtered IBC, filtering an IBC prediction of the current block from the reordered merge candidate list.

10. The method of claim 9, further comprising:

determining that the current block is reconstructed by using filtered IBC based on a syntax element or an inference without a syntax element.

11. The method of claim 8, wherein at least part of the template and at least part of the reference template are not used for obtaining the template matching score.

12. A method for decoding video data, comprising:

determining a block vector for a chroma block from a bitstream based on a luma block that is associated with the chroma block, wherein the luma block is coded with an intra block copy (IBC);

obtaining an IBC prediction of the chroma block based on the block vector;

in response to determining that a filtered IBC prediction is used for the chroma block, filtering the IBC prediction of the chroma block to obtain the filtered IBC prediction; and reconstructing the chroma block based on the filtered IBC prediction.

13. The method of claim 12, further comprising:

determining that a filtered IBC prediction is used for the chroma block based on a syntax element or a mode used by the luma block associated with the chroma block.

14. A method for encoding video data, comprising:

determining a reference block in a video frame for predicting a current block in the video frame;

obtaining a set of filter coefficients corresponding to a filter shape based on the sample values from both a template area associated with the reference block and a template area associated with the current block, wherein the template area associated with the reference block is extended based on the filter shape;

deriving, with the set of filter coefficients and the filter shape, each of predicted sample values of the current block based on a plurality of corresponding sample values associated with the reference block; and

generating a bitstream based on the predicted sample values.

15. The method of claim 14, wherein the template area associated with the reference block is extended to include one or more lines on the right of the reference block and one or more lines below the reference block.

16. The method of claim 15, wherein the template area associated with the reference block is extended to obtain an extended template area to further include one or more lines to the left of the reference block and one or more lines above the reference block.

17. The method of claim 16, wherein samples in the extended template area are padded with closest available sample values.

18. The method of claim 17, wherein the samples padded with the closest available sample values are unavailable samples.

19. The method of claim 14, wherein the template area associated with the reference block is extended to include up to N lines on the left of the reference block and up to N lines above the reference block, and N is an integer.

20. The method of claim 14, wherein the plurality of corresponding sample values associated with the reference block comprise a sample value in the extended template area.

21. A method for encoding video data, comprising:

obtaining a merge candidate list for intra block copy (IBC) prediction of a current block, the merge candidate list comprises a plurality of candidates that have been encoded with IBC;

reordering the plurality of candidates in the merge candidate list based on a template matching score of each of the plurality of candidates, the template matching score is obtained based on a difference between sample values of a template of a candidate and corresponding reference sample values of a reference template of a reference block of that candidate, wherein in response to determining that a candidate is coded with filtered IBC, the corresponding reference sample values of the reference template are obtained using filtered IBC; and

generating a bitstream by encoding the current block based on the reordered merge candidate list.

22. The method of claim 21, further comprising:

filtering an IBC prediction of the current block from the reordered merge candidate list.

23. The method of claim 22, further comprising sending a syntax element in the bitstream to indicate that filtered IBC is used for the current block.

24. The method of claim 21, wherein at least part of the template and at least part of the reference template are not used for obtaining the template matching score.

25. A method for encoding video data, comprising:

determining a block vector for a chroma block based on a luma block that is associated with the chroma block, wherein the luma block is coded with an intra block copy (IBC);

obtaining an IBC prediction of the chroma block based on the block vector;

filtering the IBC prediction of the chroma block; and

generating a bitstream based on the filtered IBC prediction.

26. The method of claim 25, further comprising sending a syntax element in the bitstream

to indicate that filtered IBC is used for the chroma block.

27. An apparatus, comprising:

one or more processors; and

one or more storage devices storing computer-executable instructions that, when executed, cause the one or more processors to perform the operations of the method of any of claims 1-26.

28. A computer program product, storing computer-executable instructions that, when executed, cause one or more processors to perform the operations of the method of any of claims 1-26.

29. A computer readable storage medium storing instructions which when executed by a computing device having one or more processors, cause the one or more processors to perform the decoding method according to any of claims 1-13 and storing a bitstream to be decoded by the decoding method according to any of claims 1-13.

30. A computer readable storage medium storing instructions which when executed by a computing device having one or more processors, cause the one or more processors to perform the encoding method according to any of claims 14-26 and storing a bitstream generated by the encoding method according to any of claims 14-26.

31. A computer readable medium storing a bitstream, wherein the bitstream is to be decoded by performing the operations of the method of any of claims 1-13.

32. A computer readable medium storing a bitstream, wherein the bitstream is obtained by performing the operations of the method of any of claims 14-26.

33. A method for receiving a bitstream to be decoded by the decoding method according to any of claims 1-13.

34. A method for transmitting a bitstream generated by the encoding method according to any of claims 14-26.

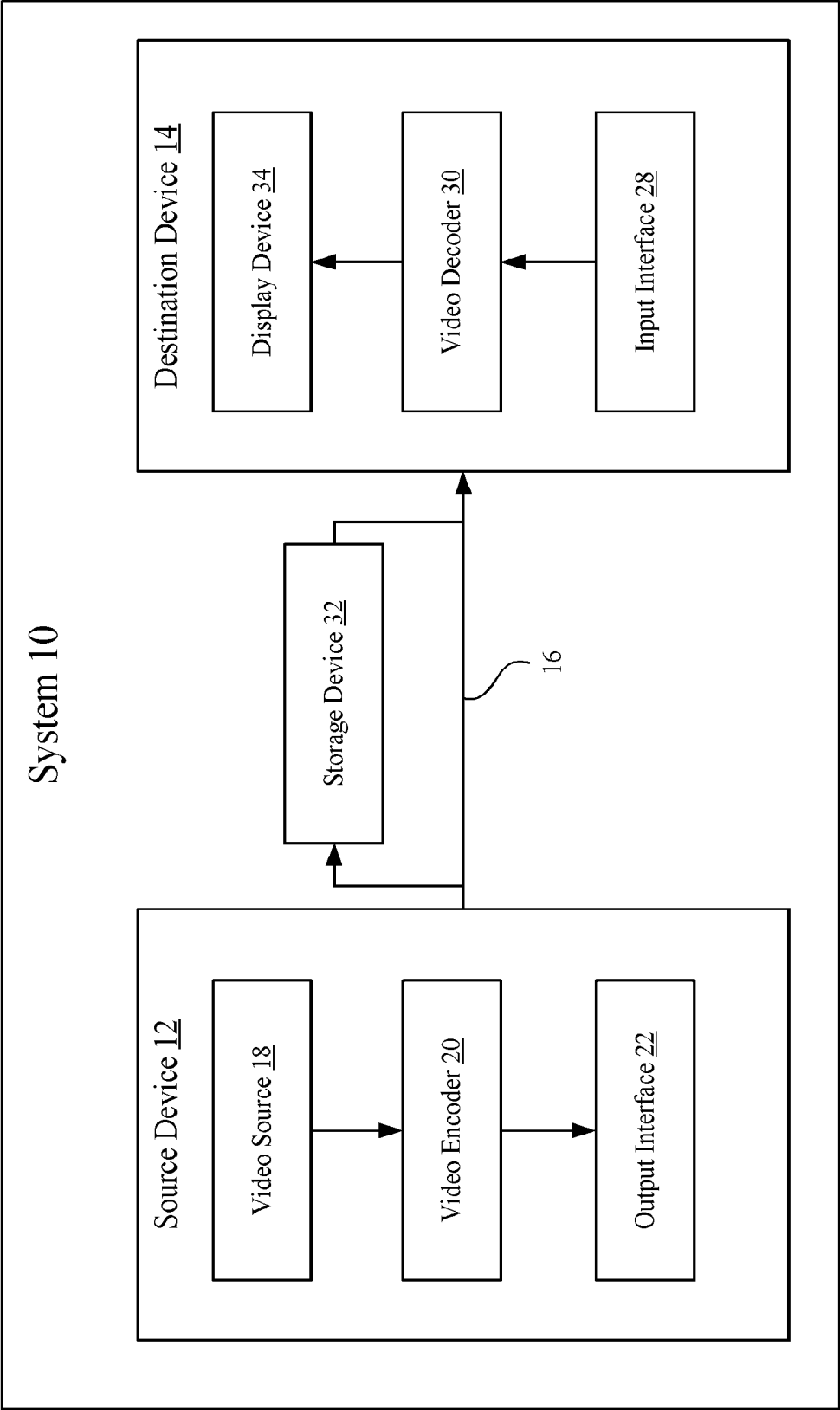


Figure 1

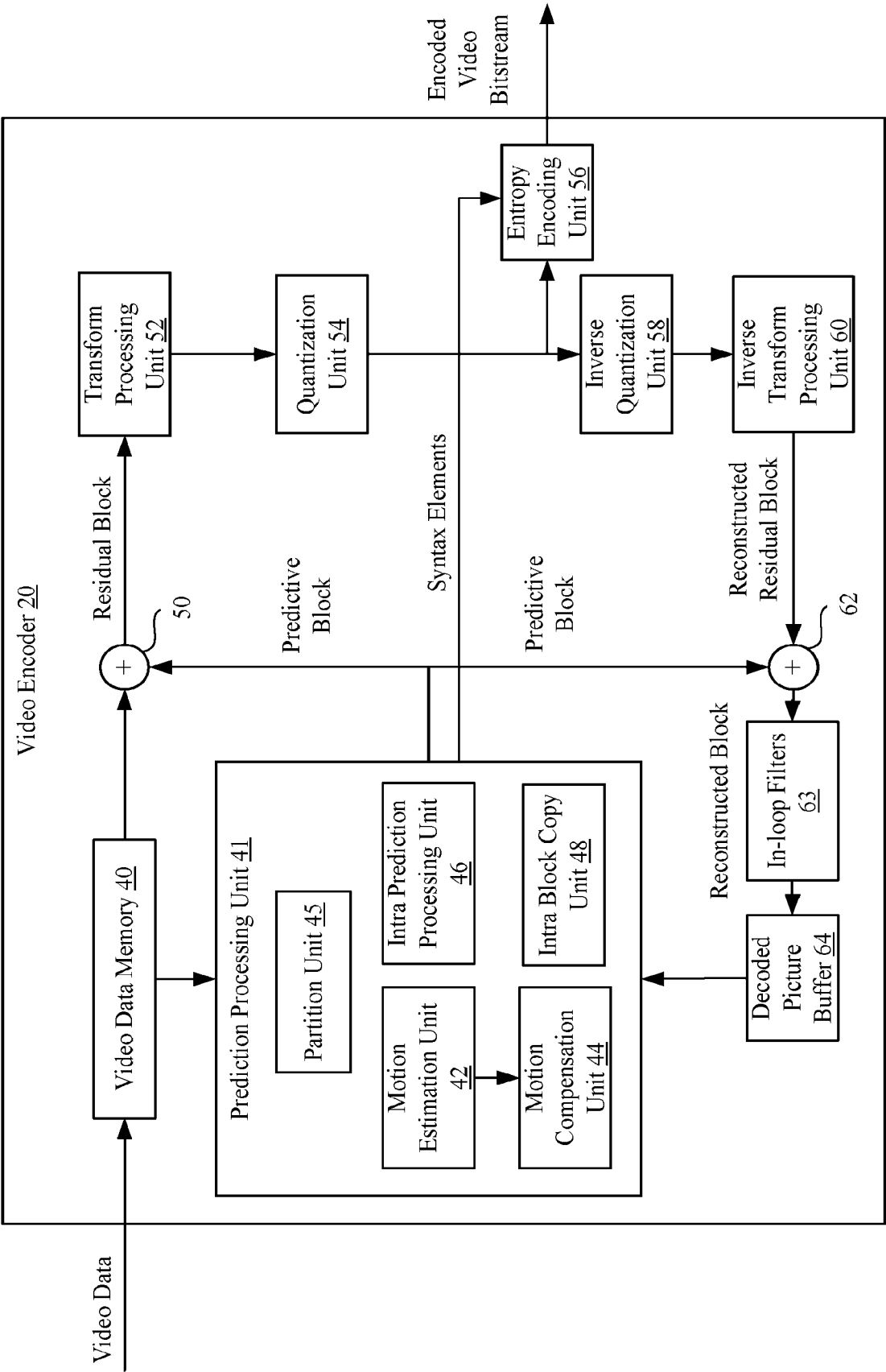


Figure 2

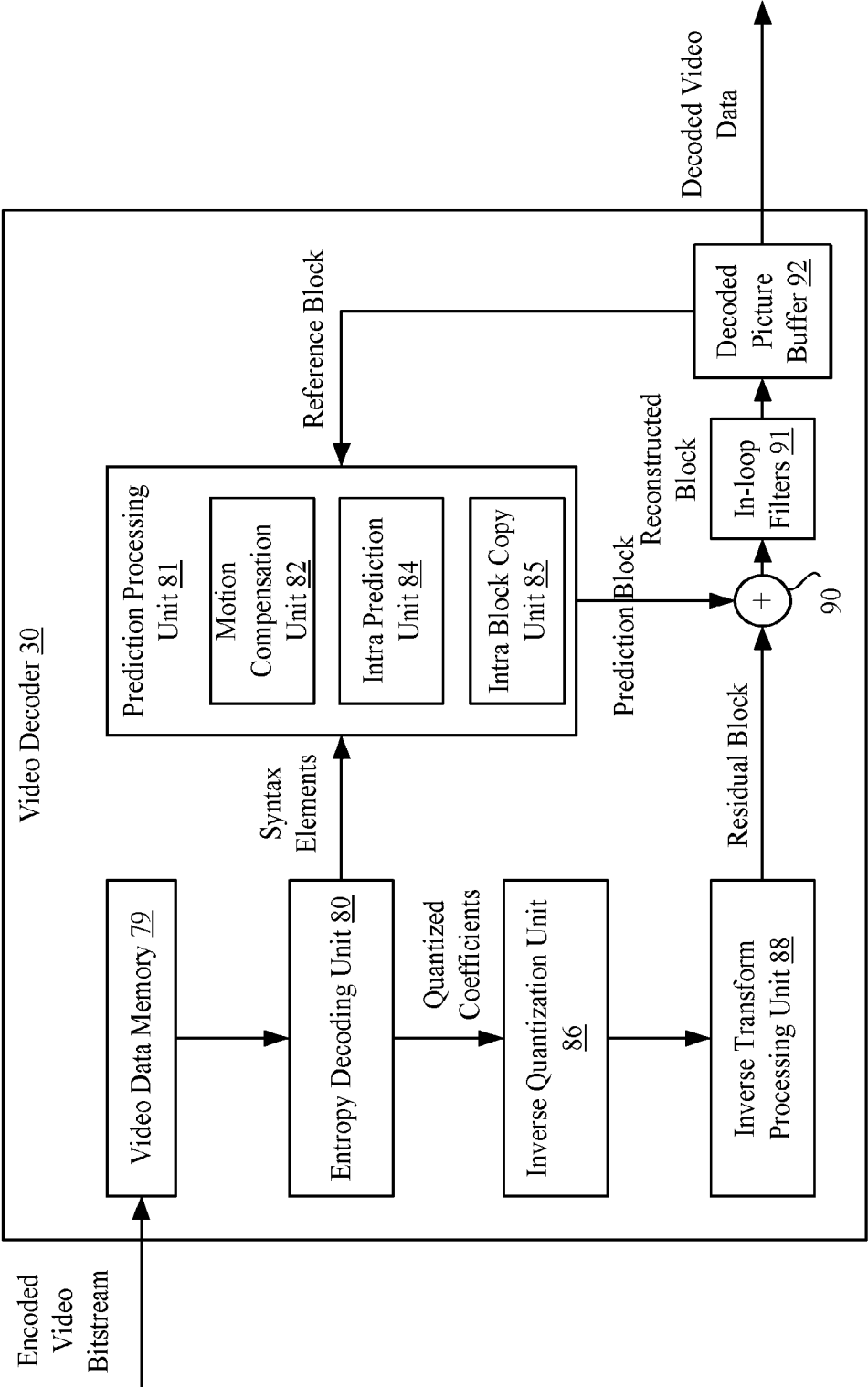


Figure 3

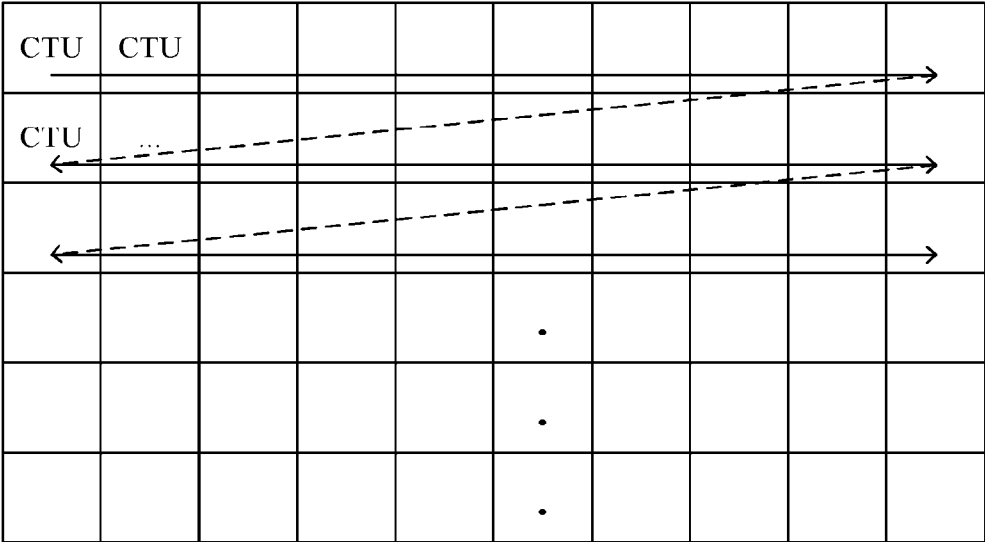


Figure 4A

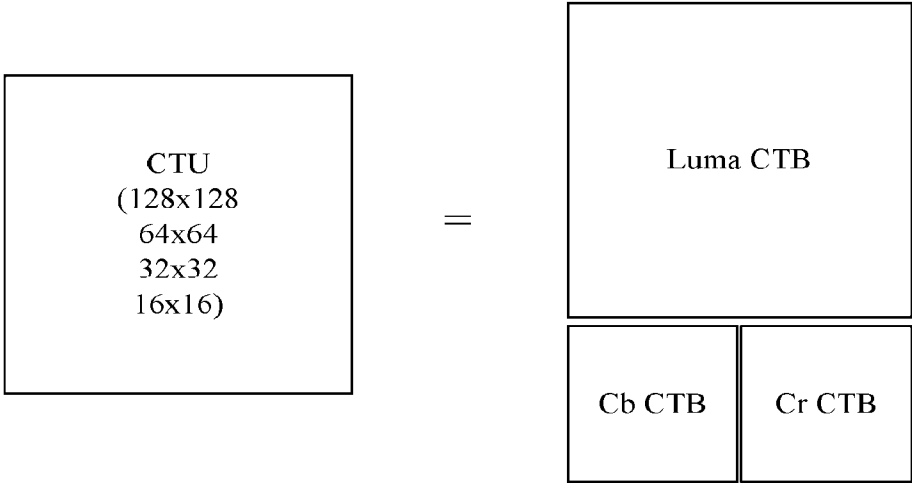


Figure 4B

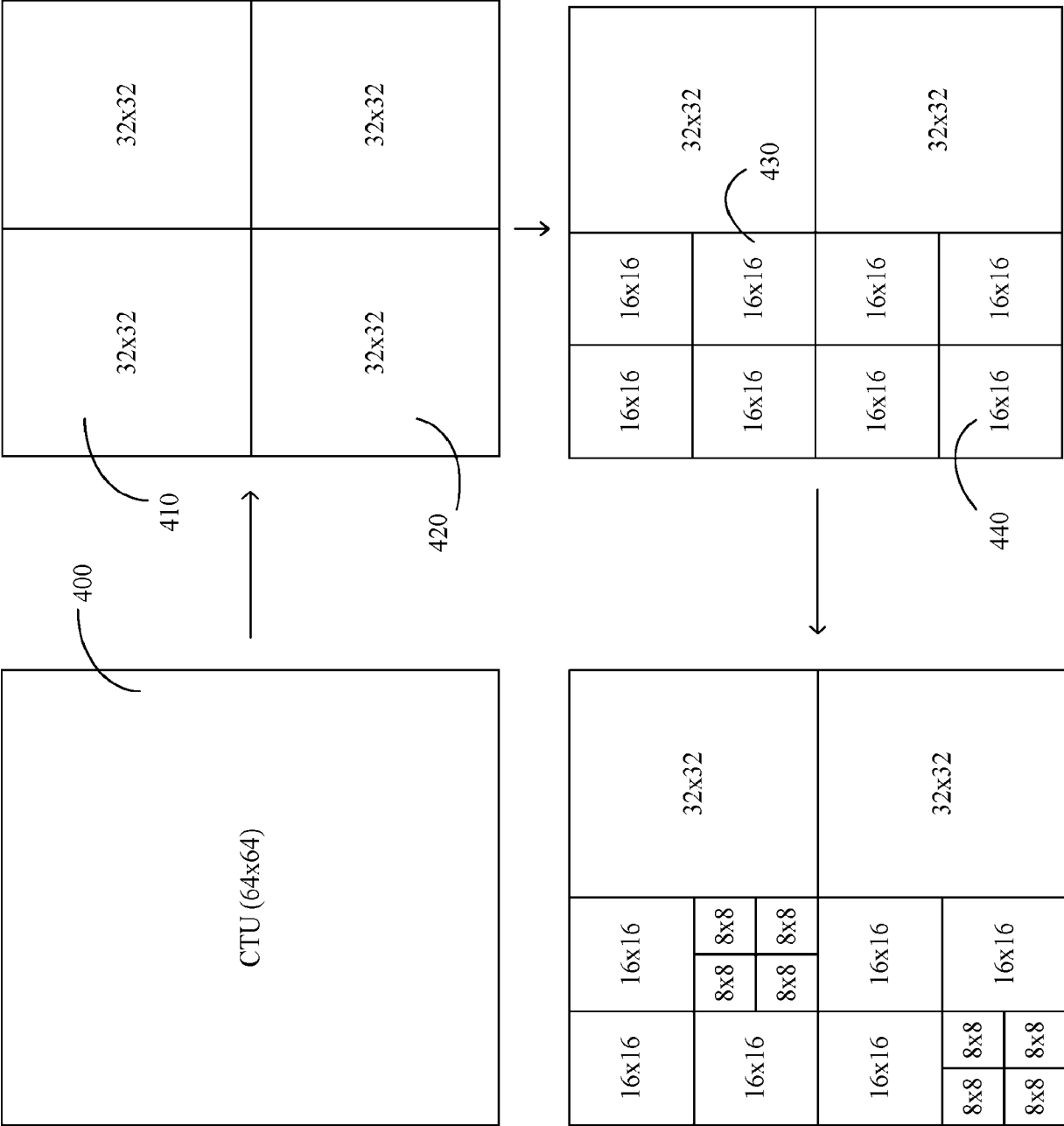


Figure 4C

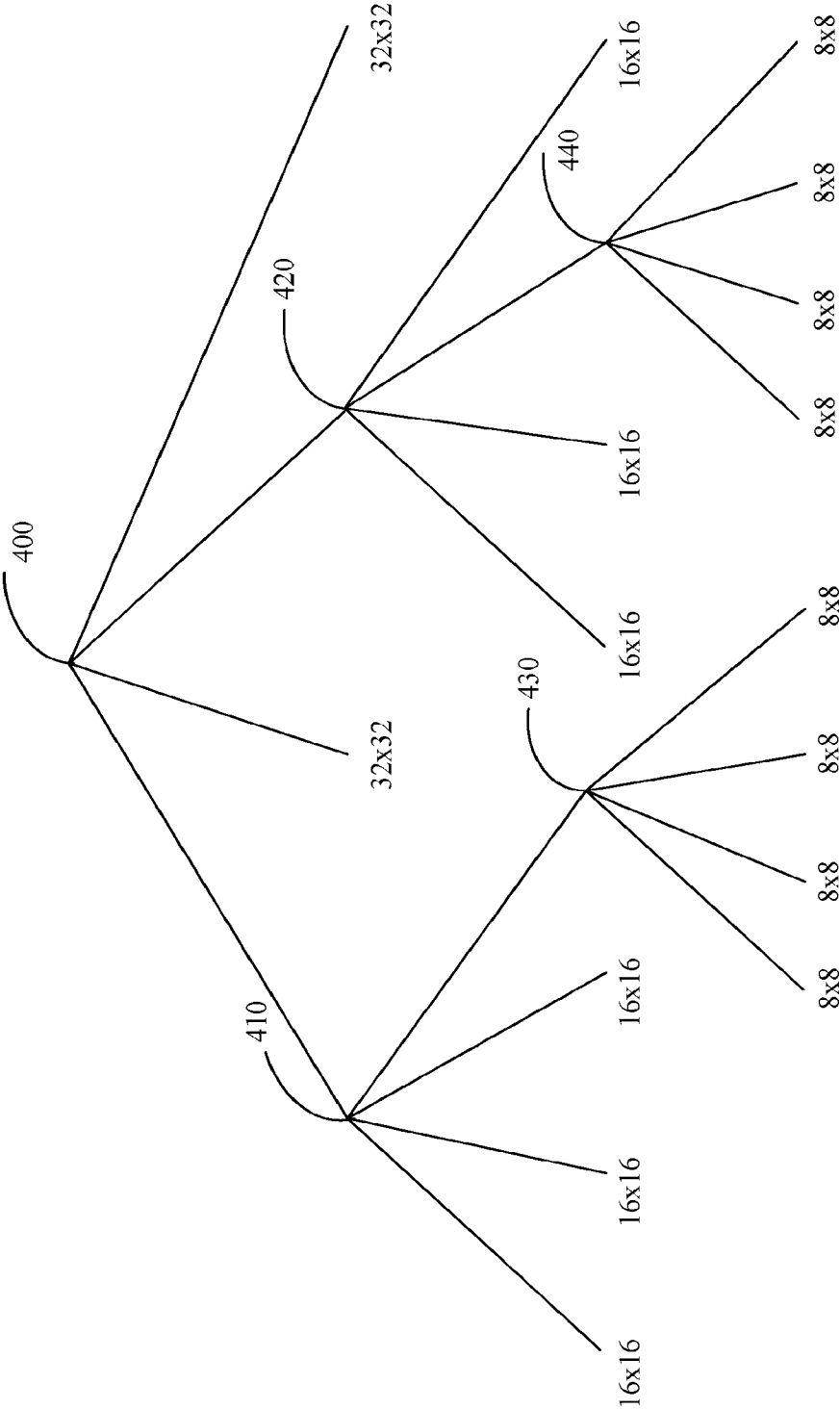


Figure 4D

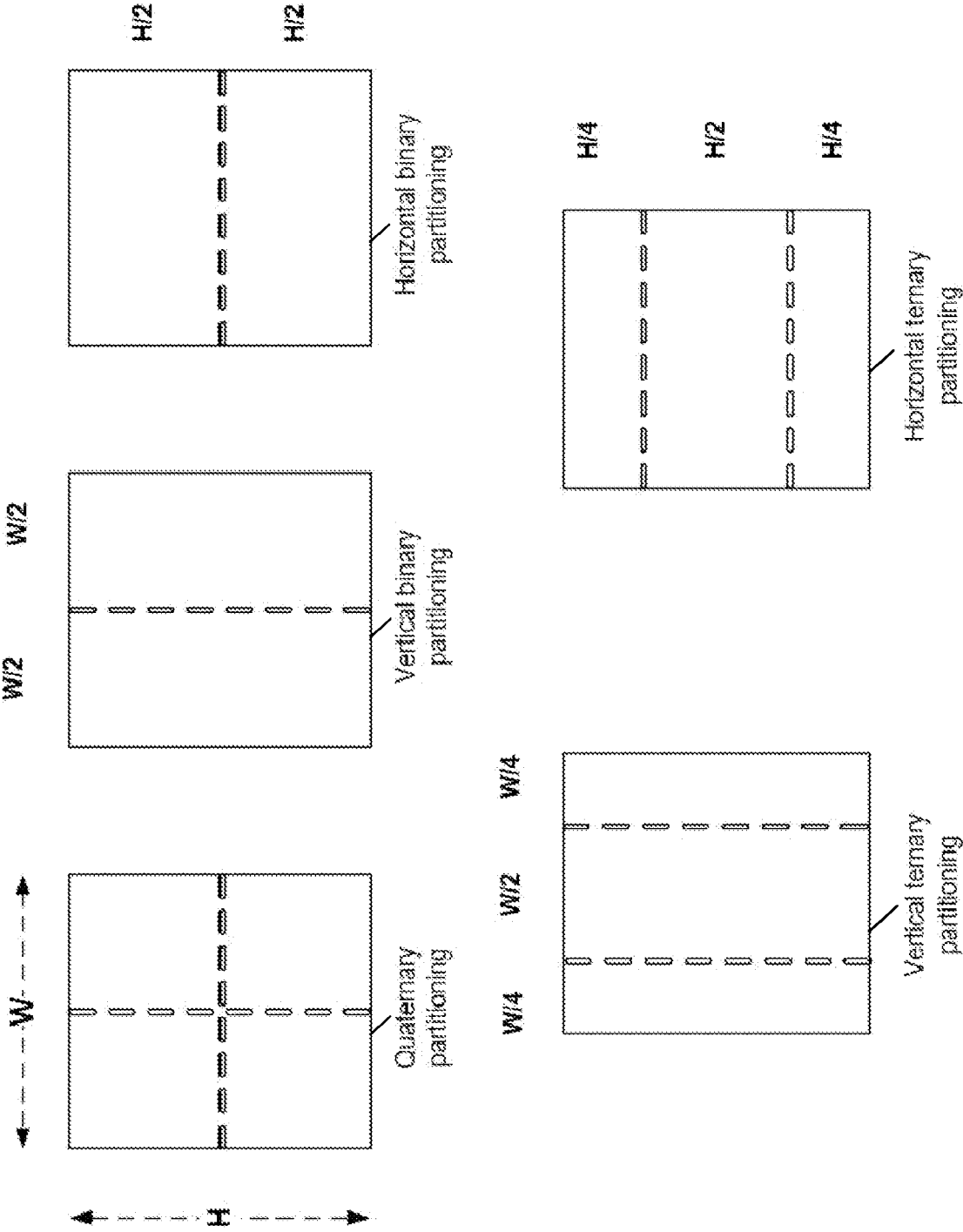


Figure 4E

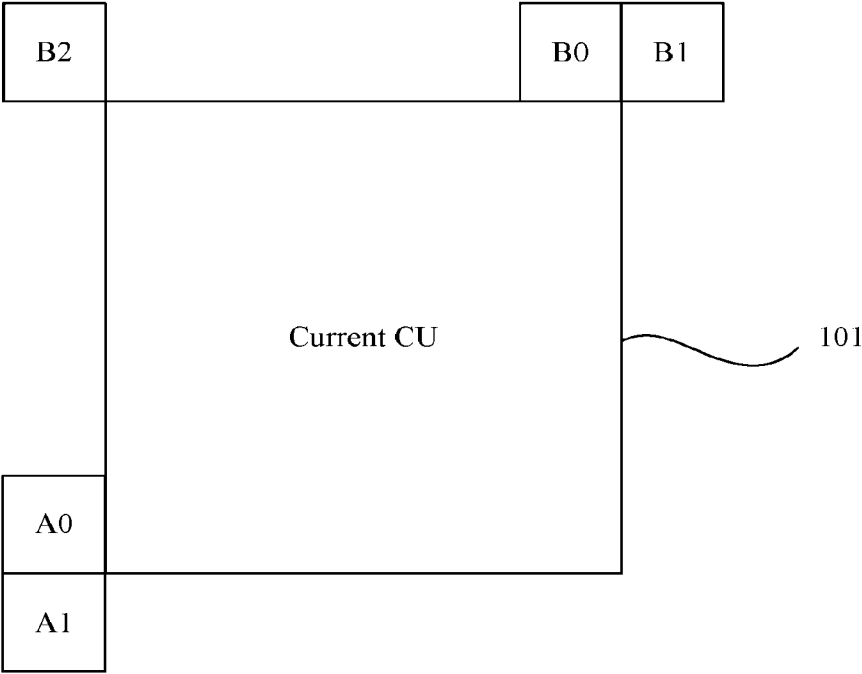


Figure 5

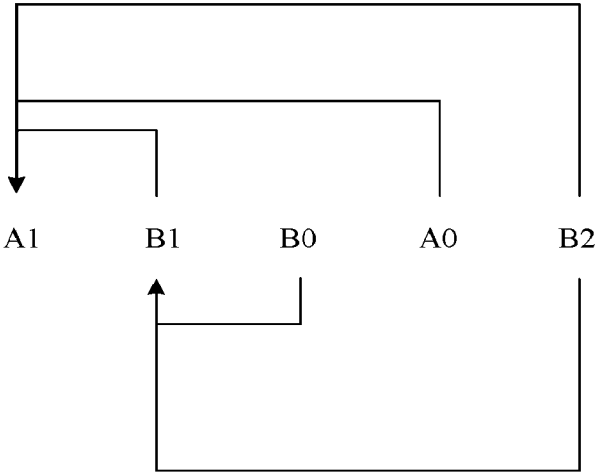


Figure 6

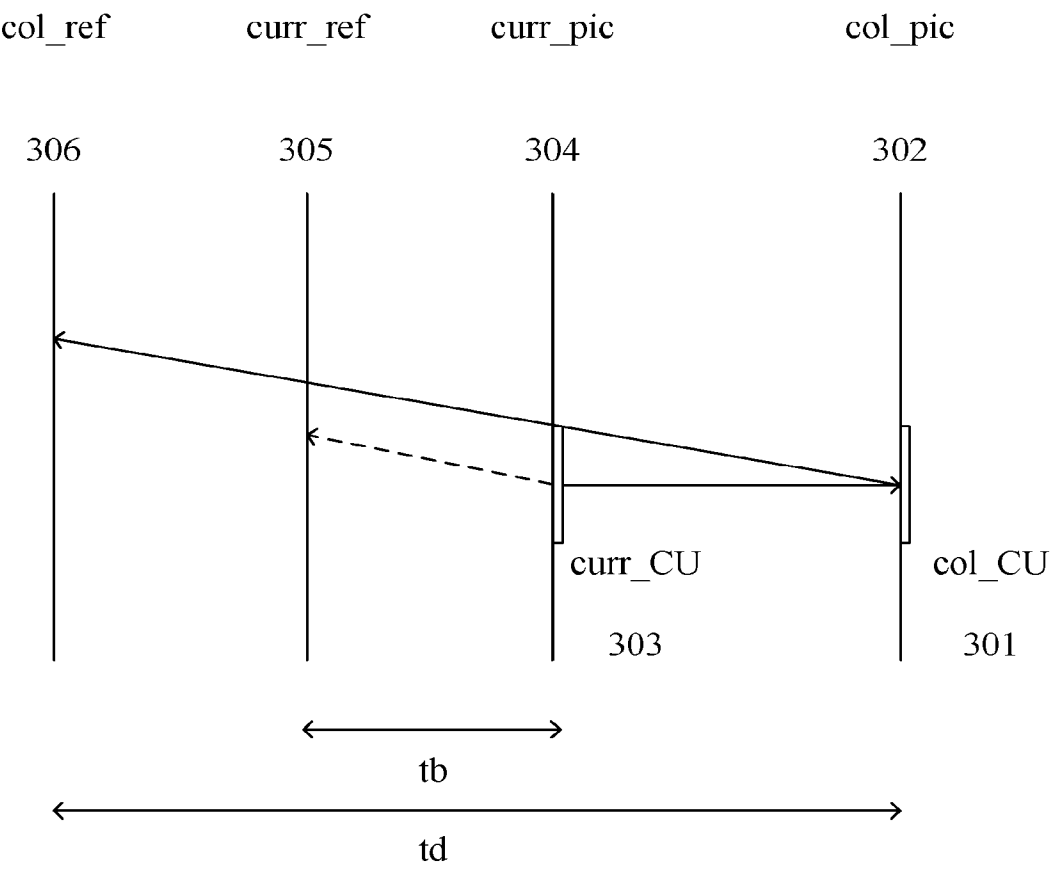


Figure 7

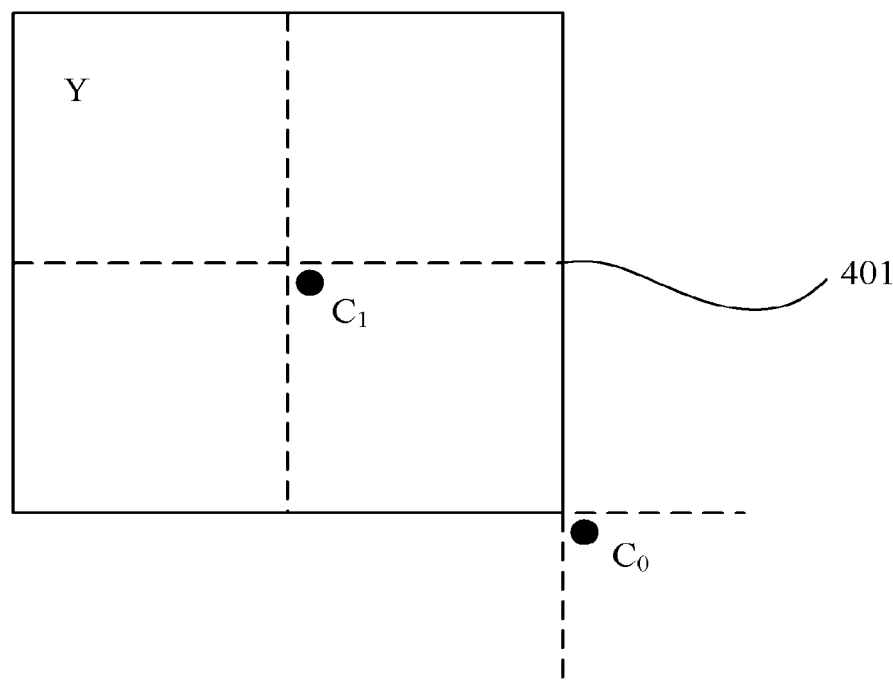


Figure 8

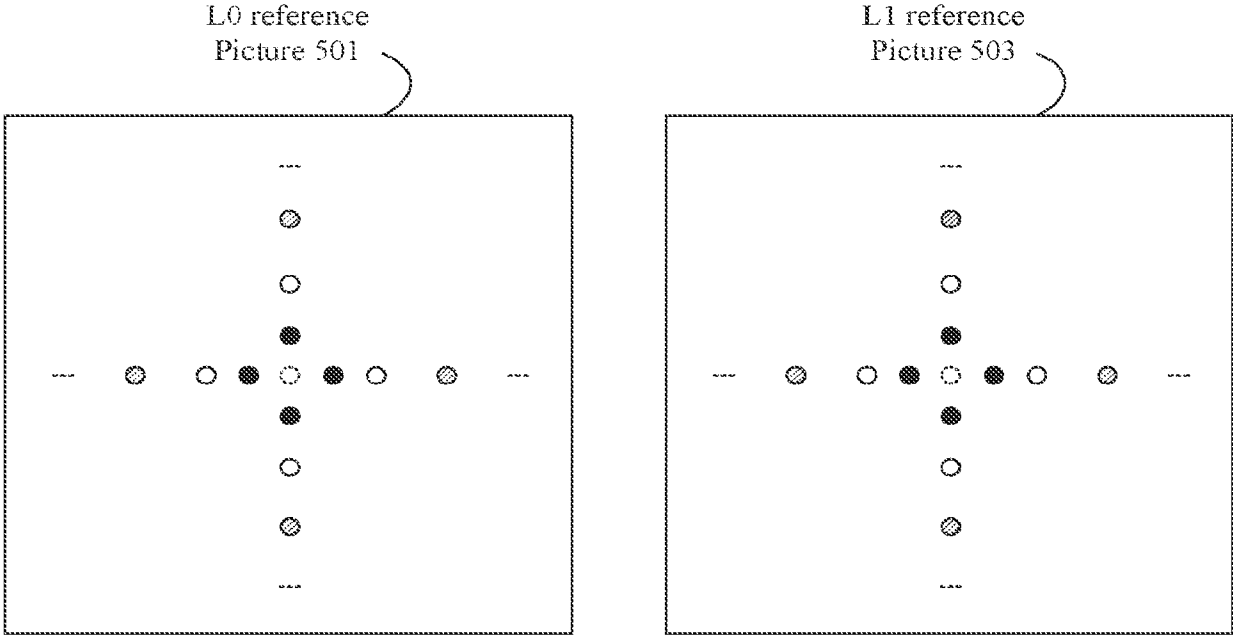


Figure 9

	L0 MV	L1 MV
Merge Index		
0	×	
1		×
2	×	
3		×
4	×	

Figure 10

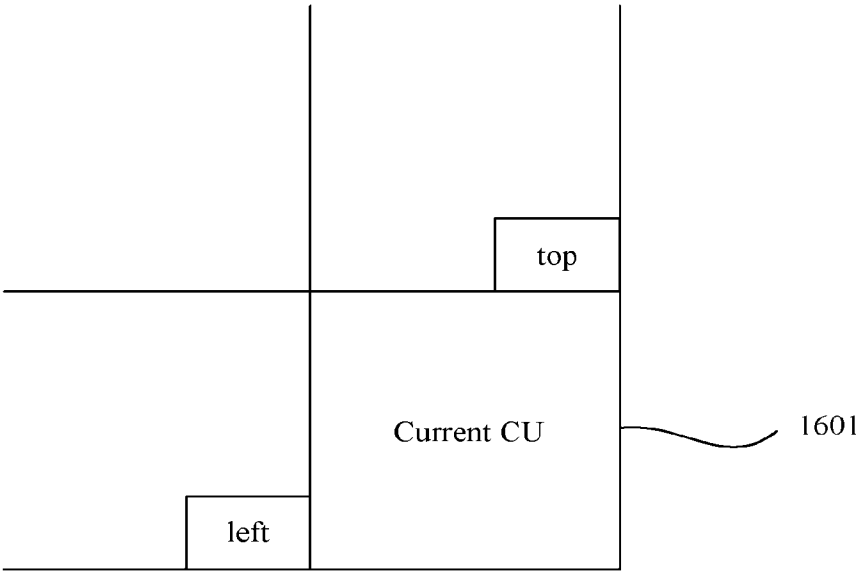


Figure 11

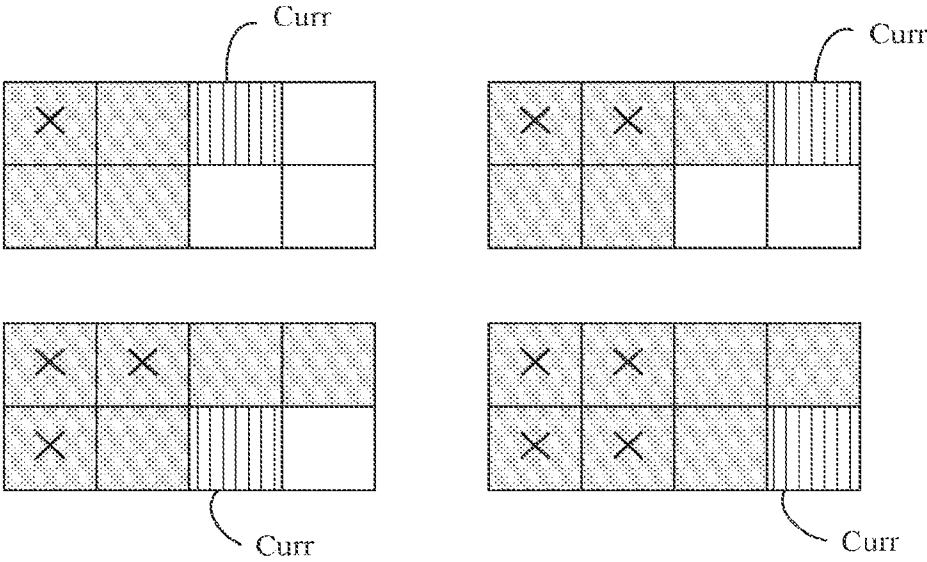


Figure 12

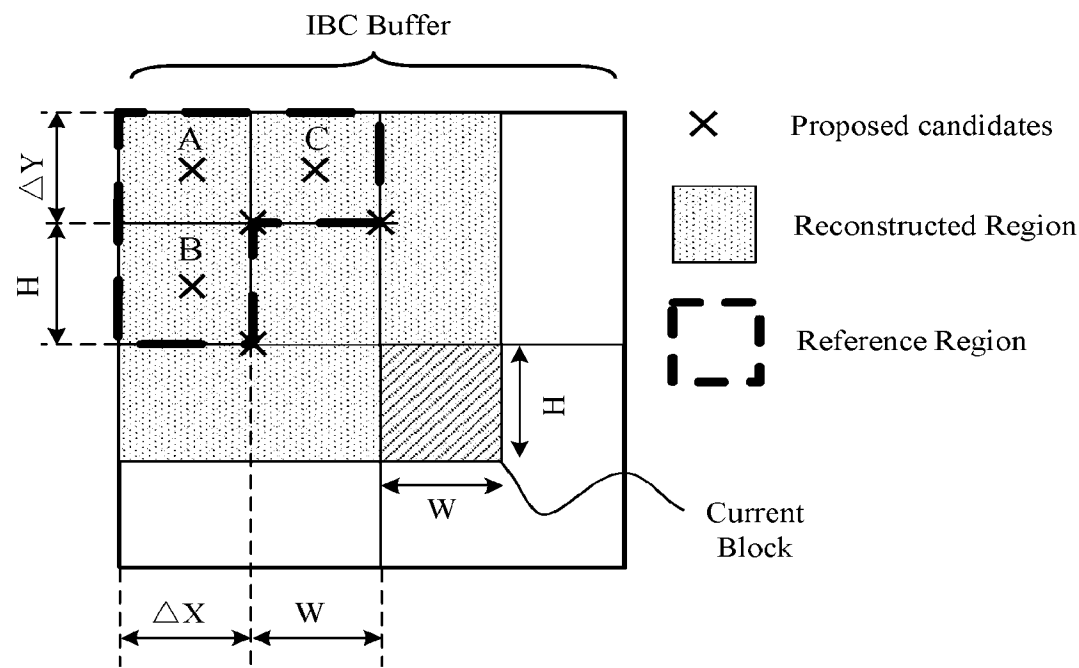
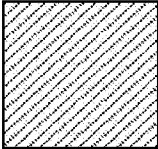
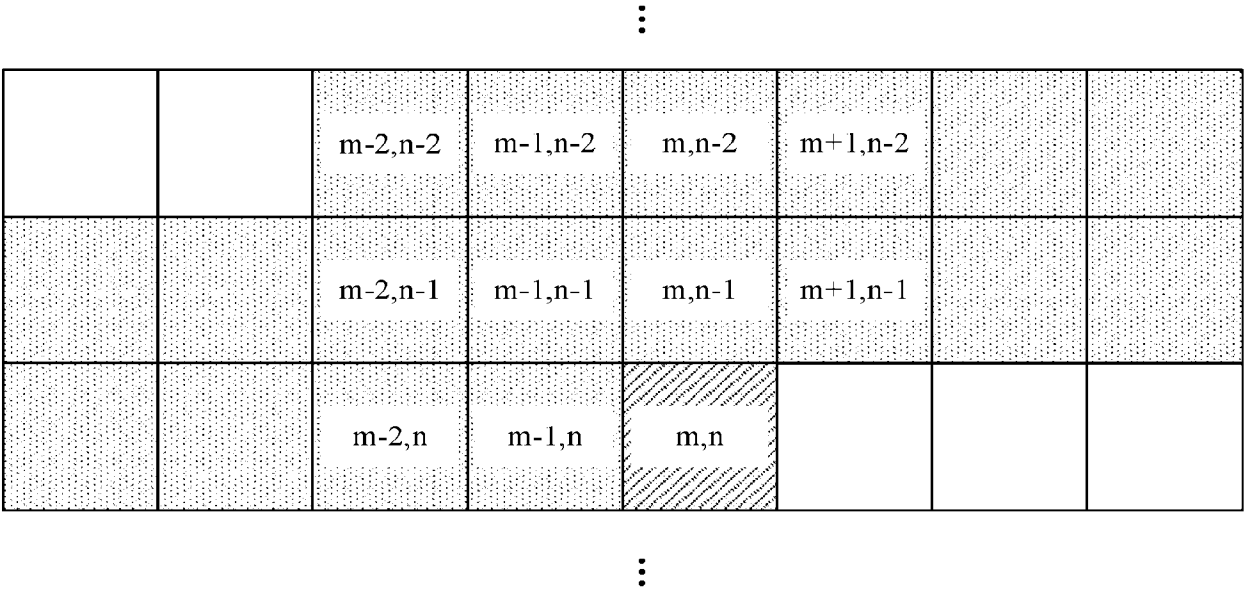
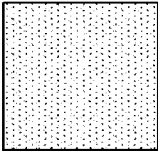


Figure 13



Current CTU



Reference area



Invalid reference area

Figure 14

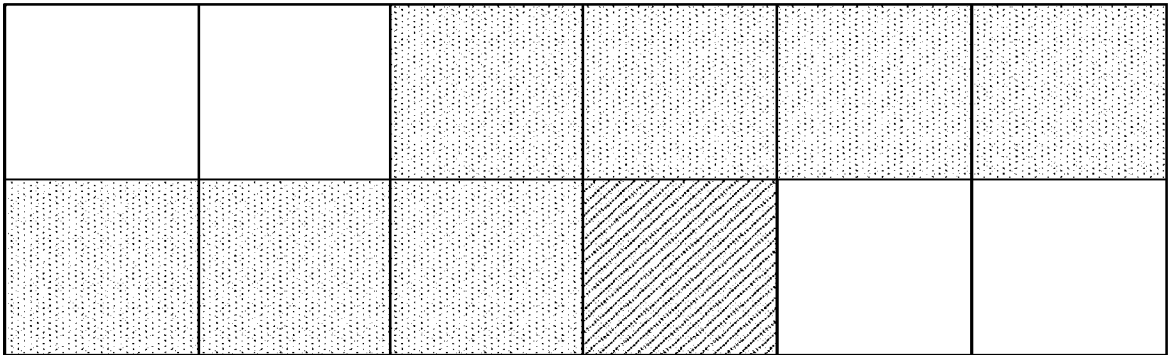


Figure 15

0	1	2	3
---	---	---	---

Figure 16A

0
1
2
3

Figure 16B

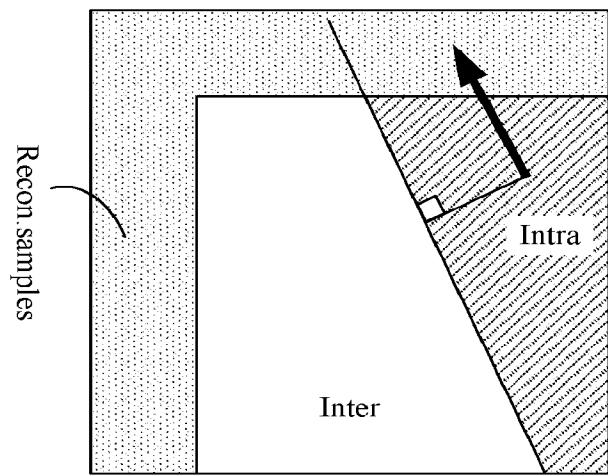


Figure 17A

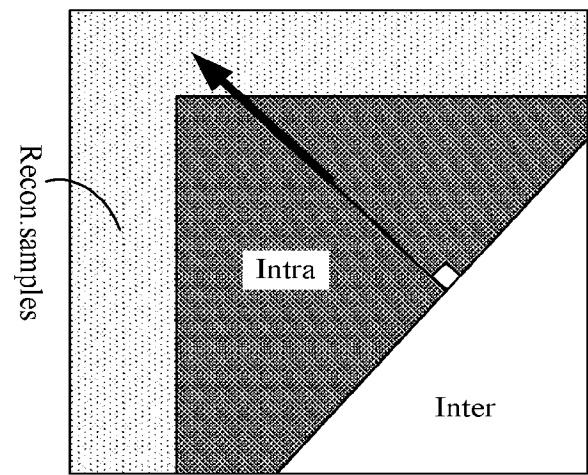


Figure 17B

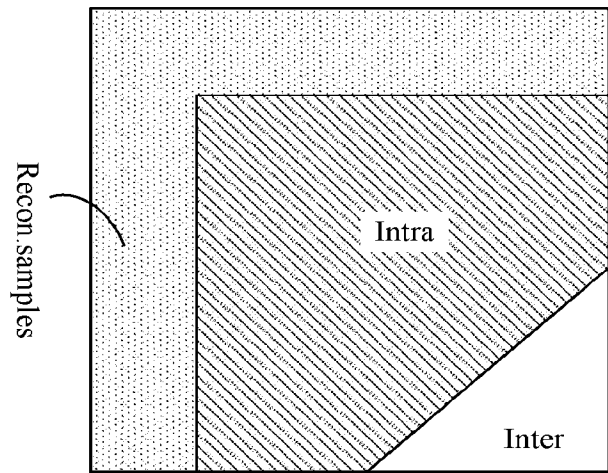


Figure 17C

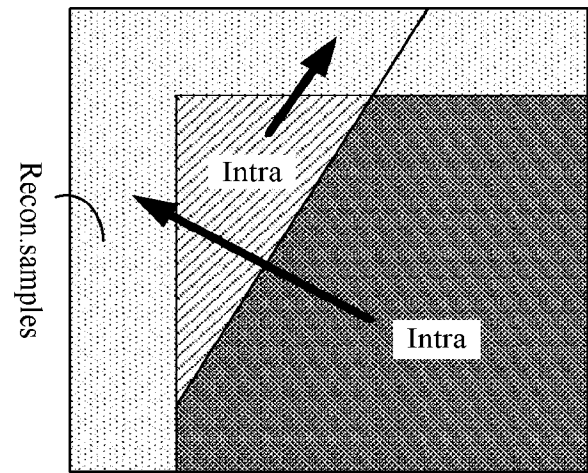


Figure 17D

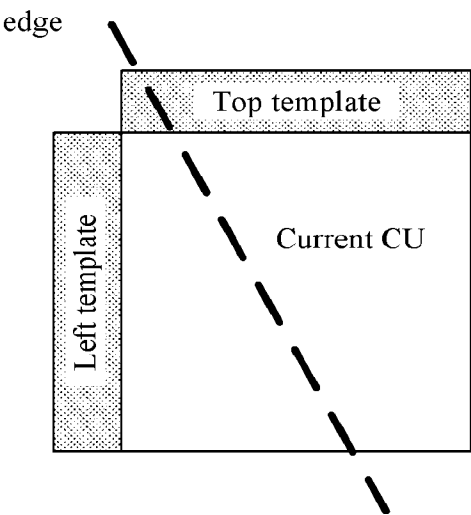


Figure 18

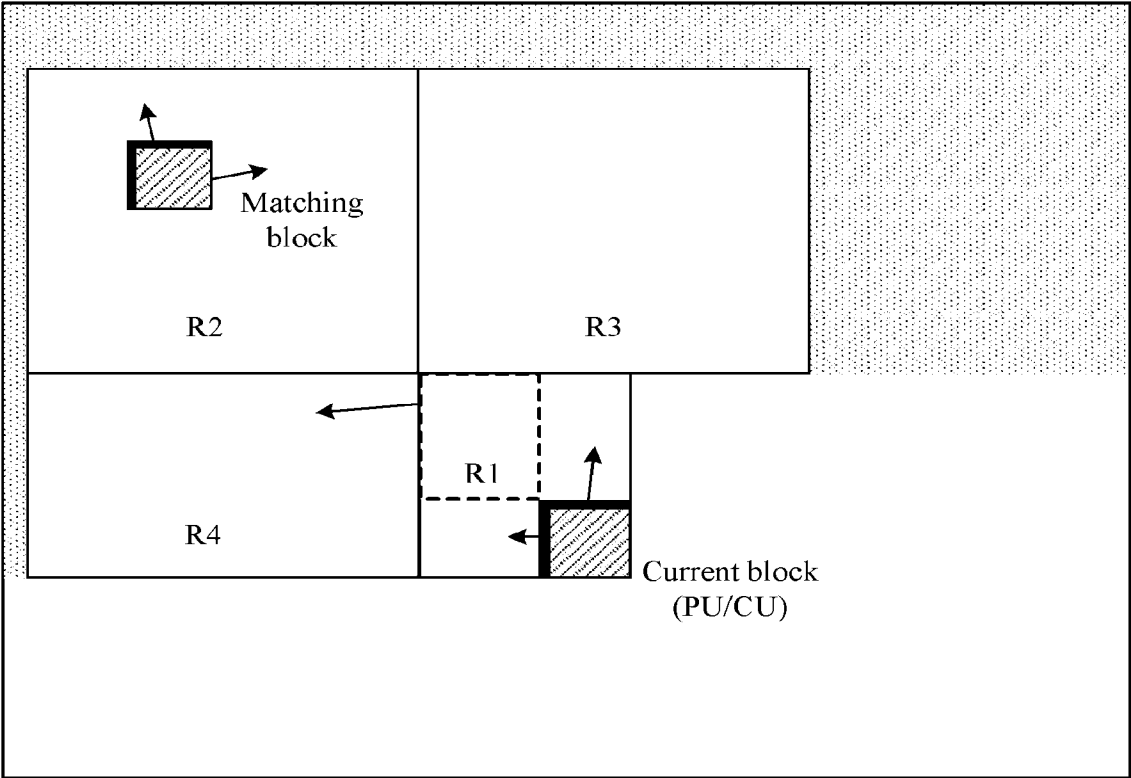


Figure 19

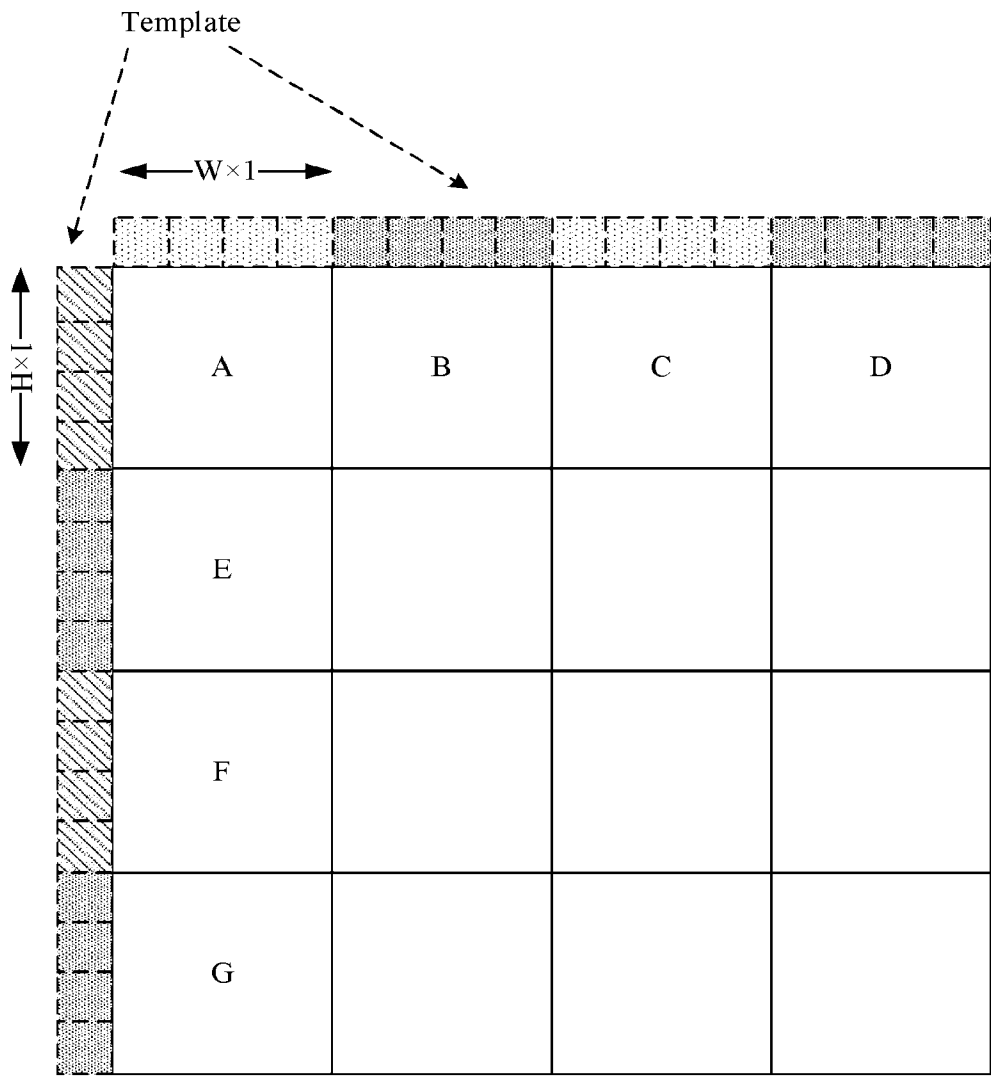


Figure 20

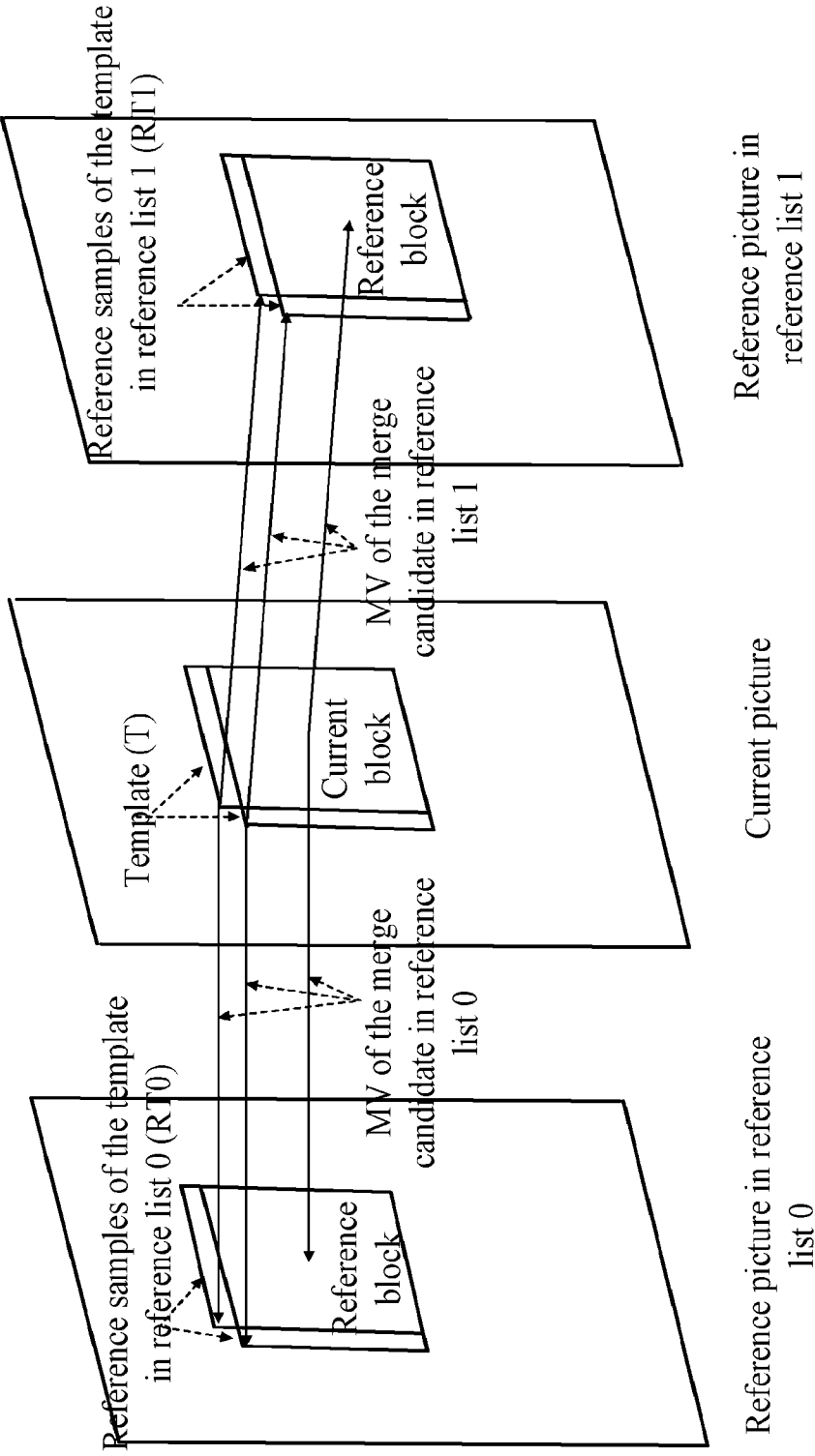


Figure 21

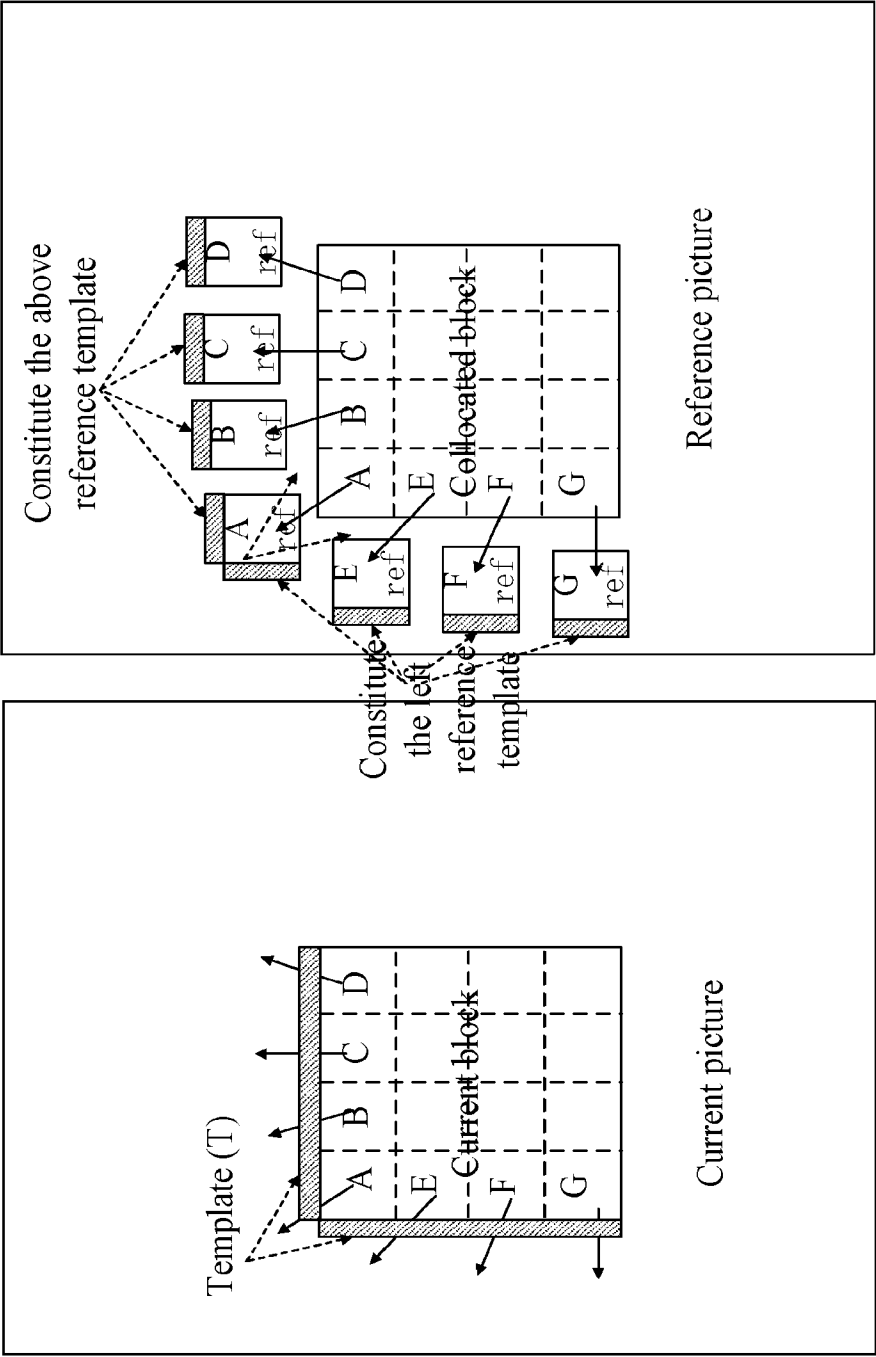


Figure 22

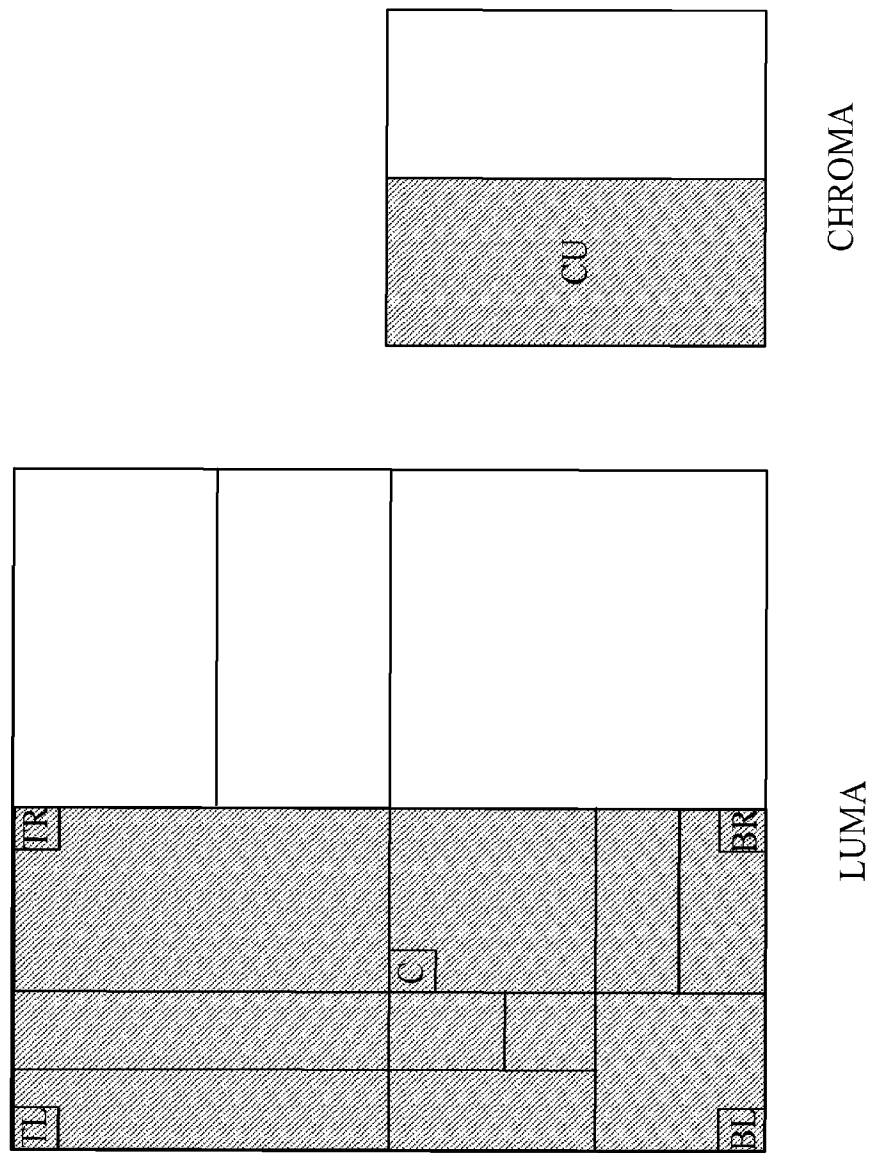


Figure 23

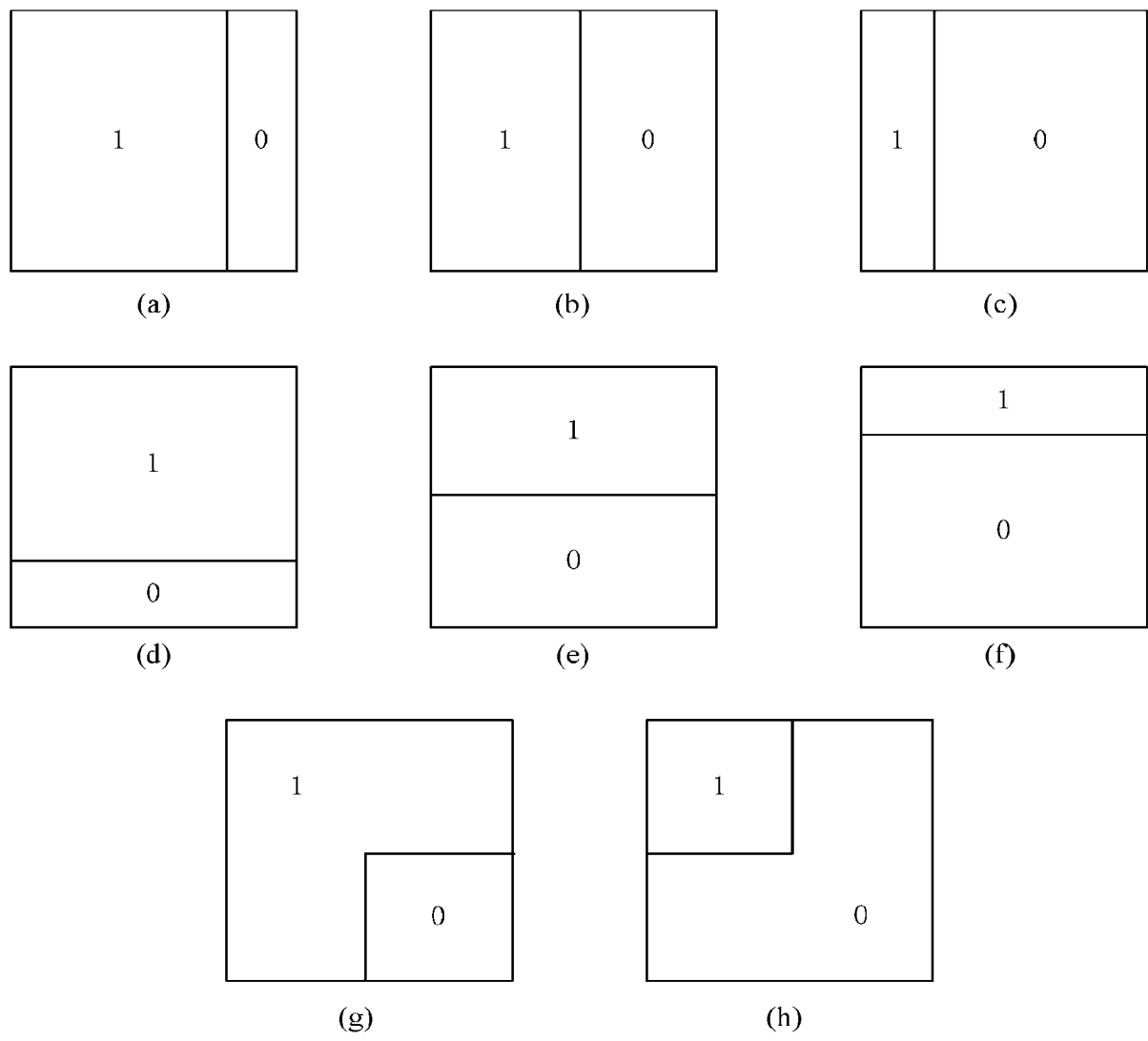


Figure 24

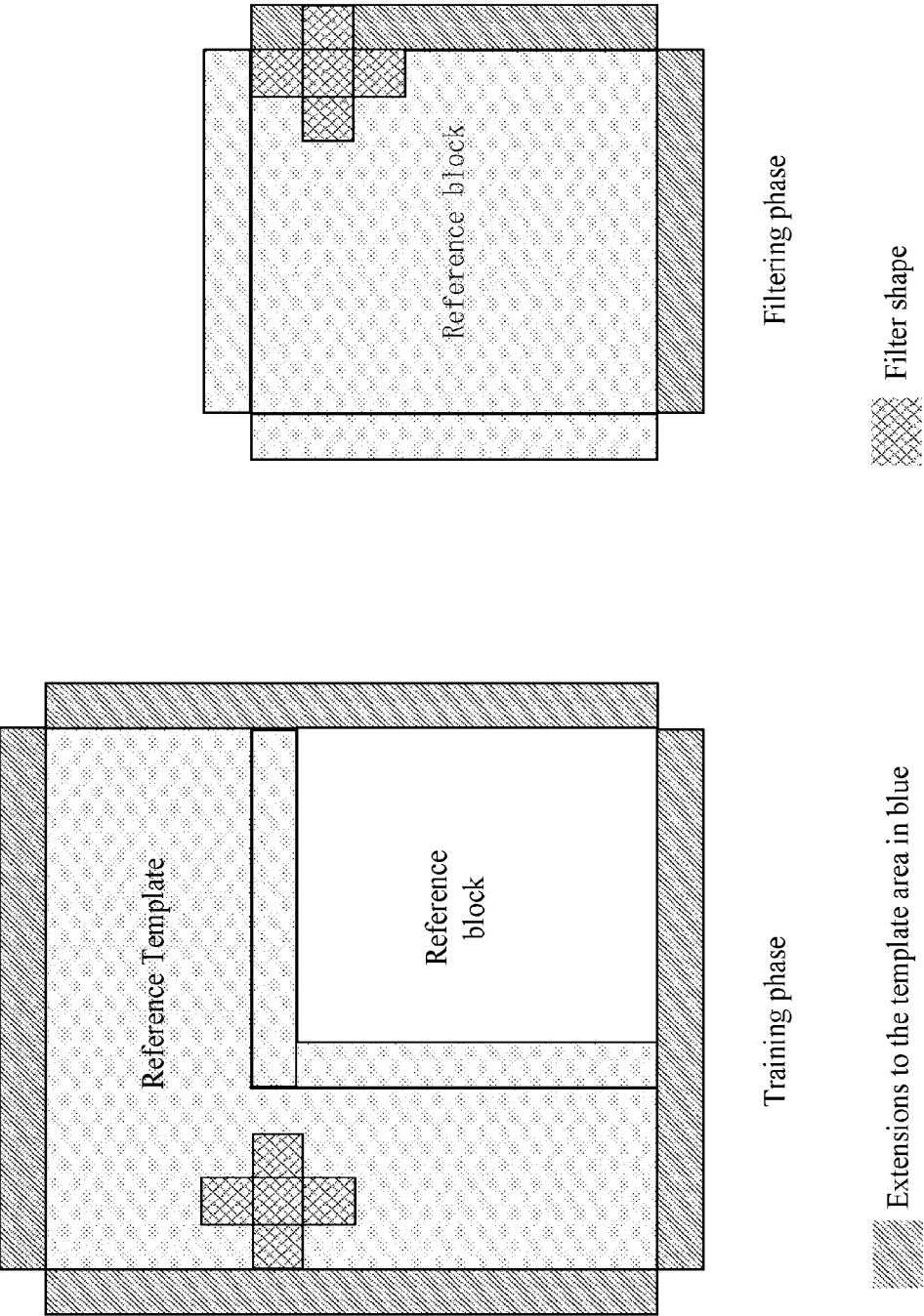


Figure 25

L ₀	L ₁	L ₂
L ₃	L ₄	L ₅
L ₆	L ₇	L ₈

Figure 26

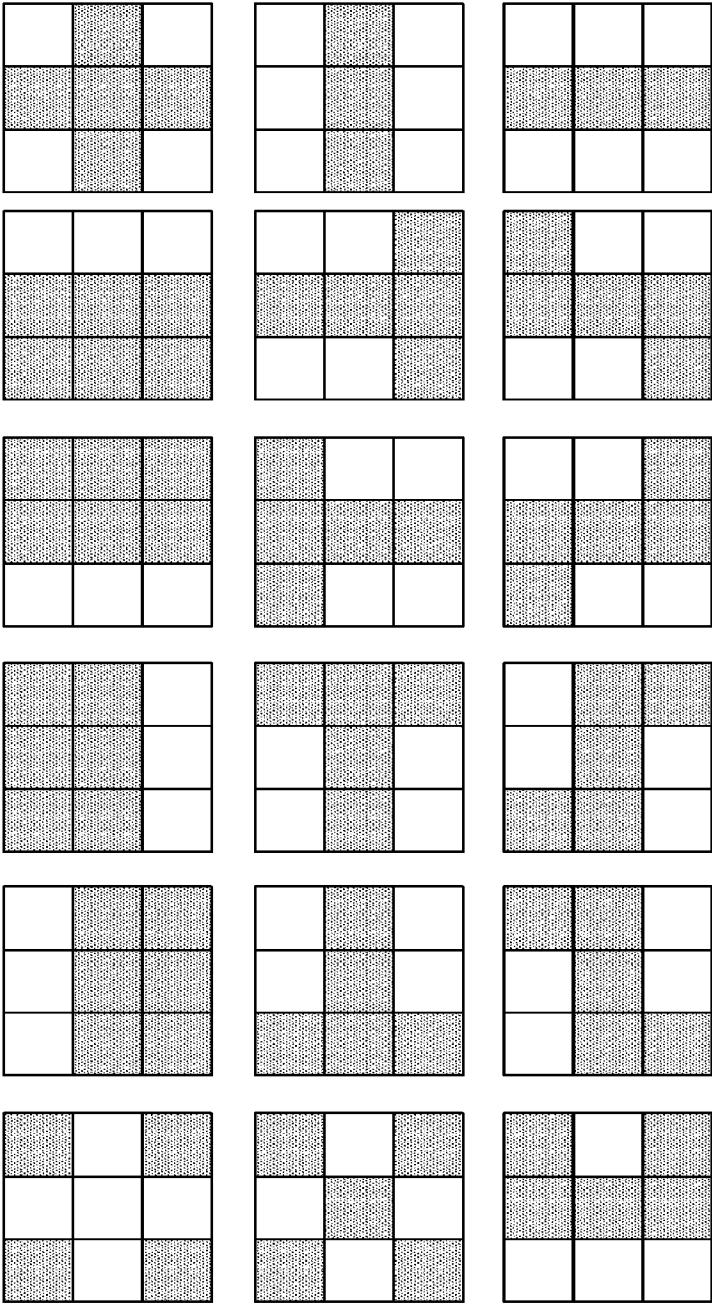


Figure 27

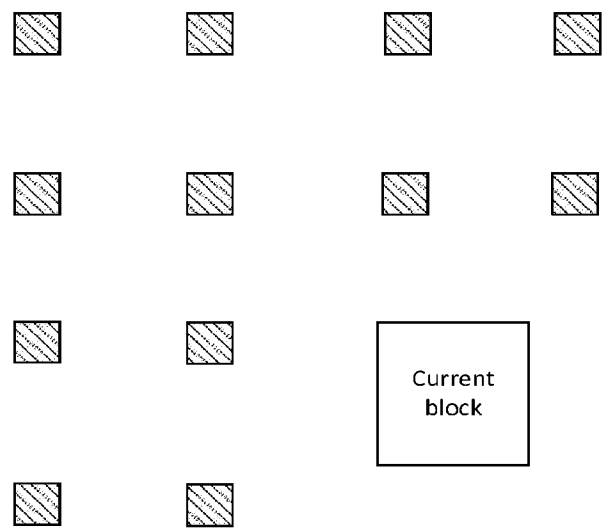


Figure 28

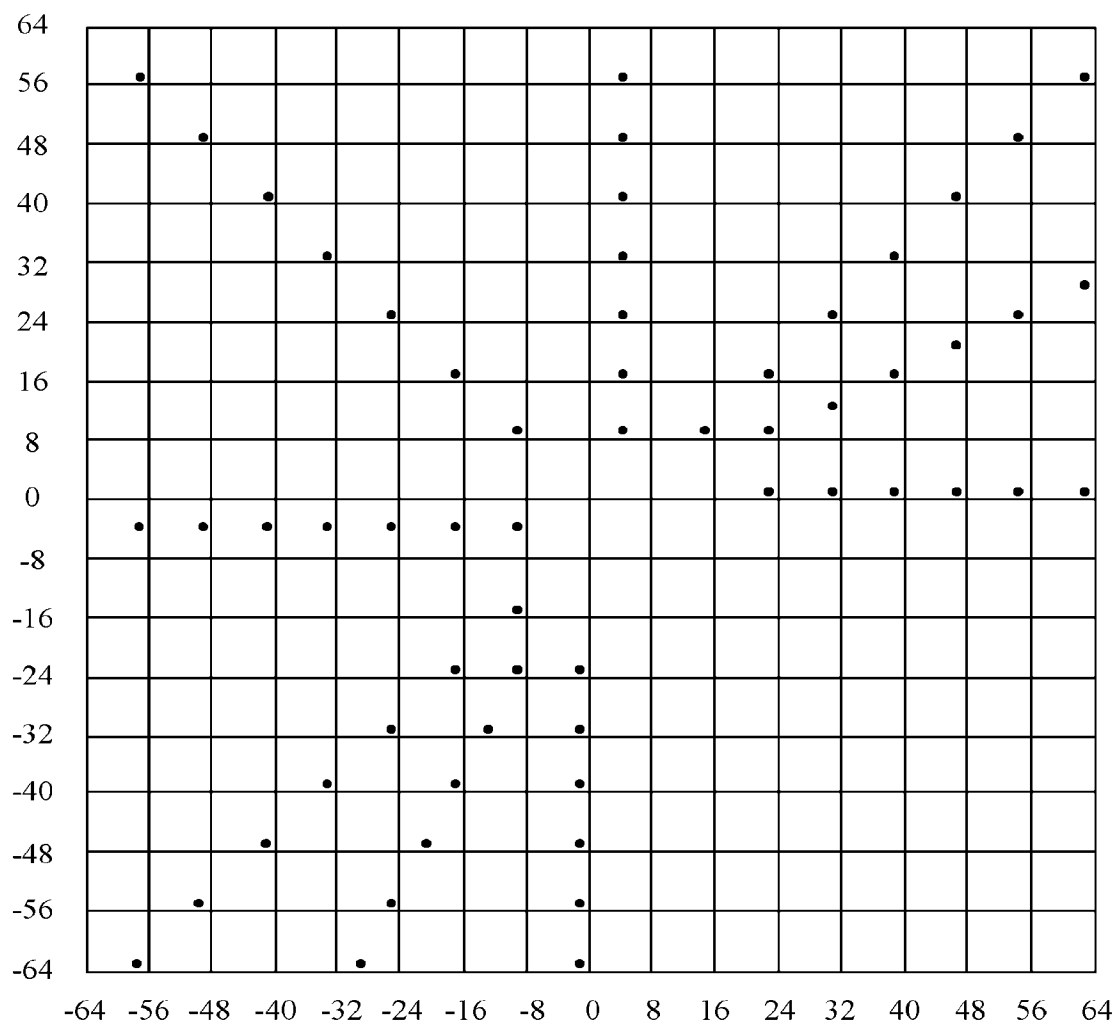


Figure 29

33/44

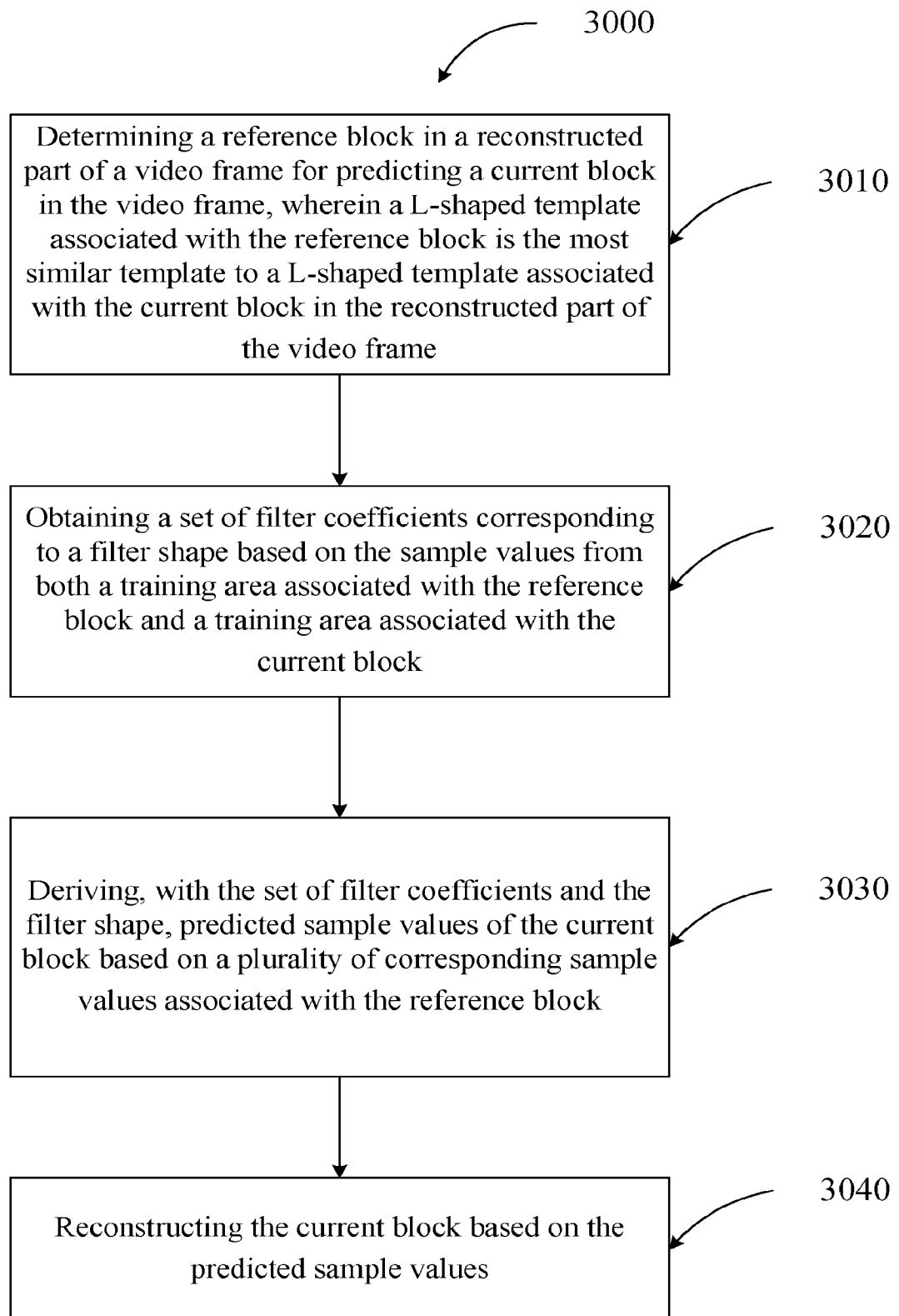


Figure 30

34/44

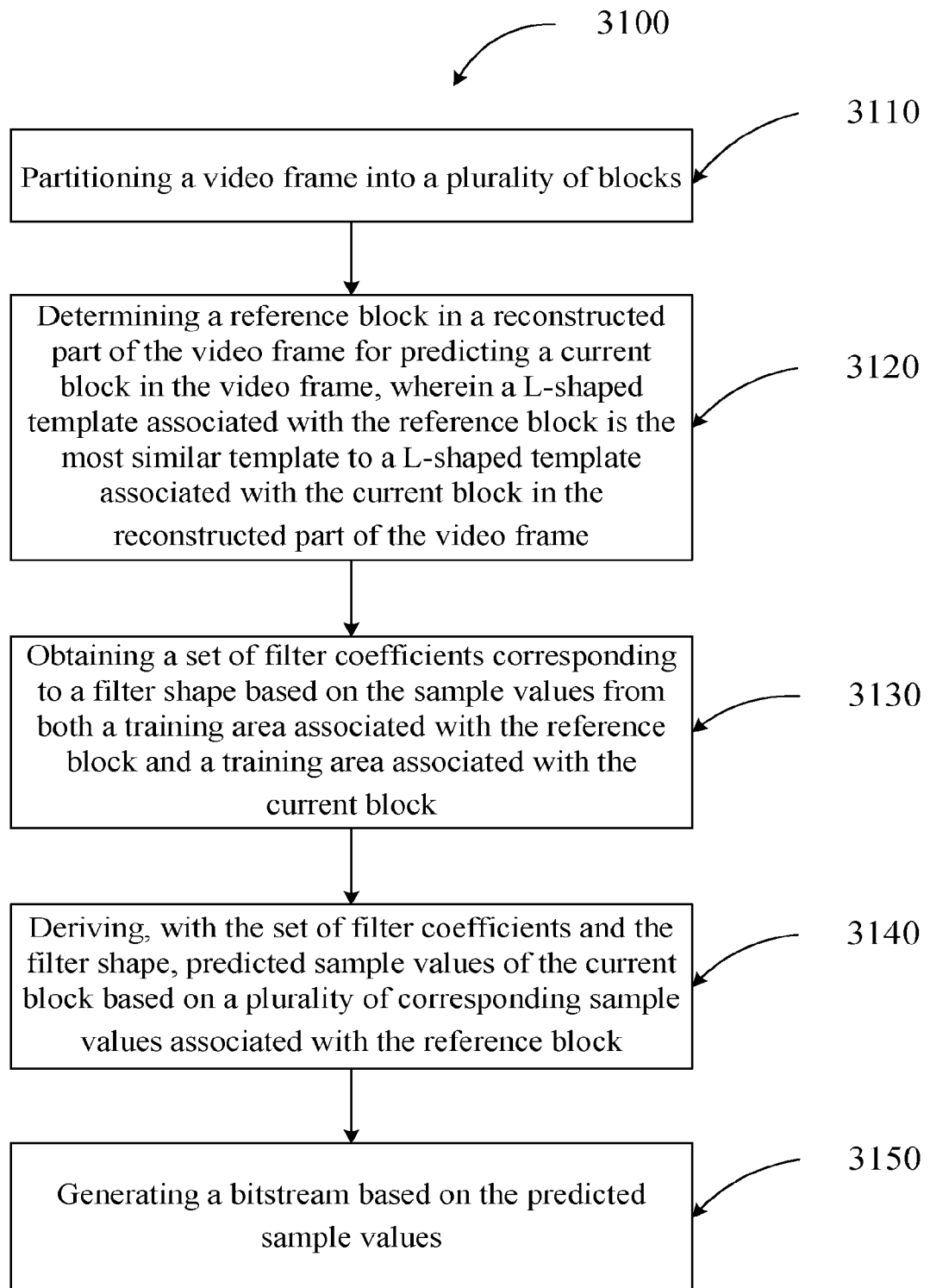


Figure 31

35/44

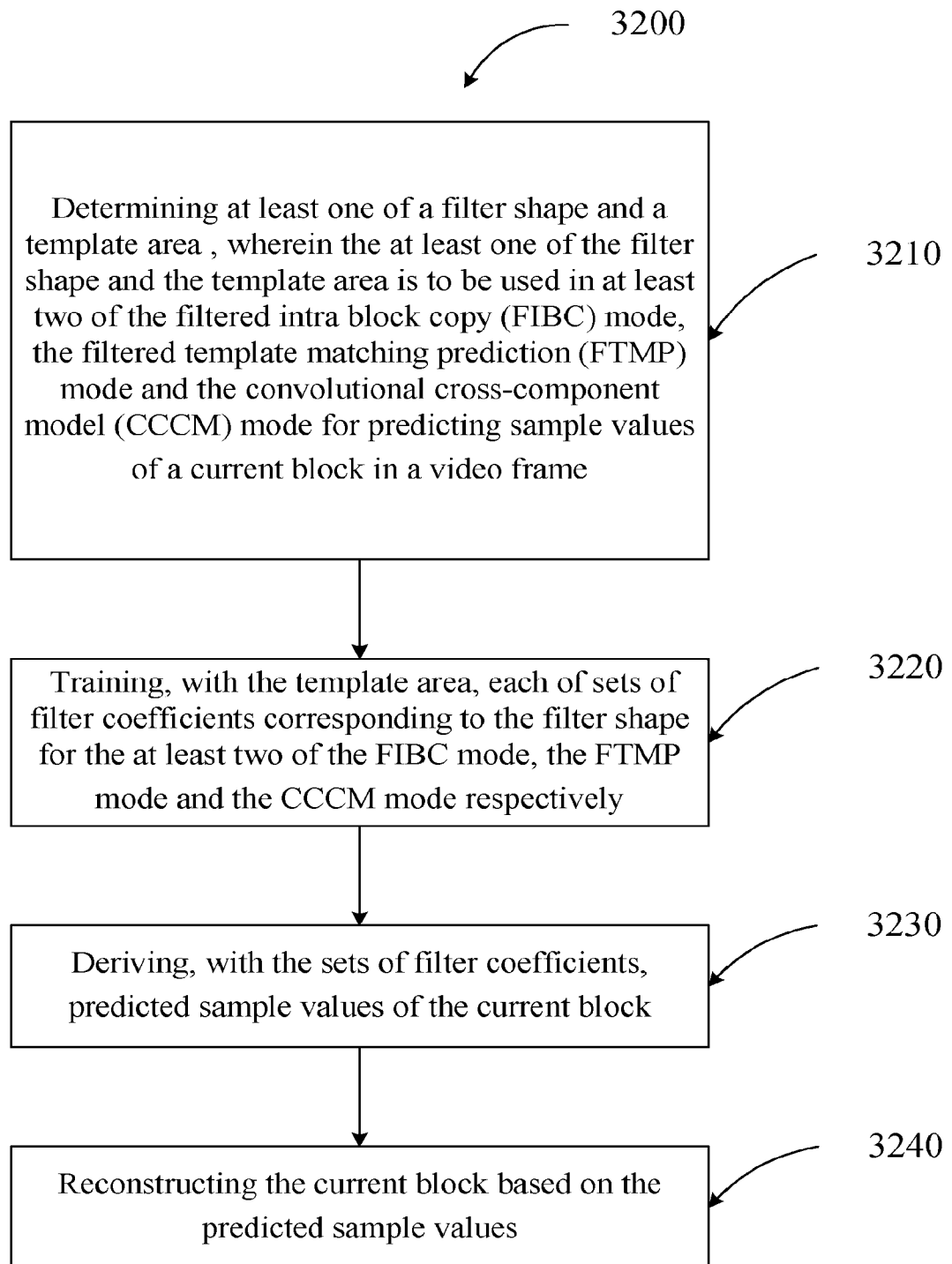


Figure 32

36/44

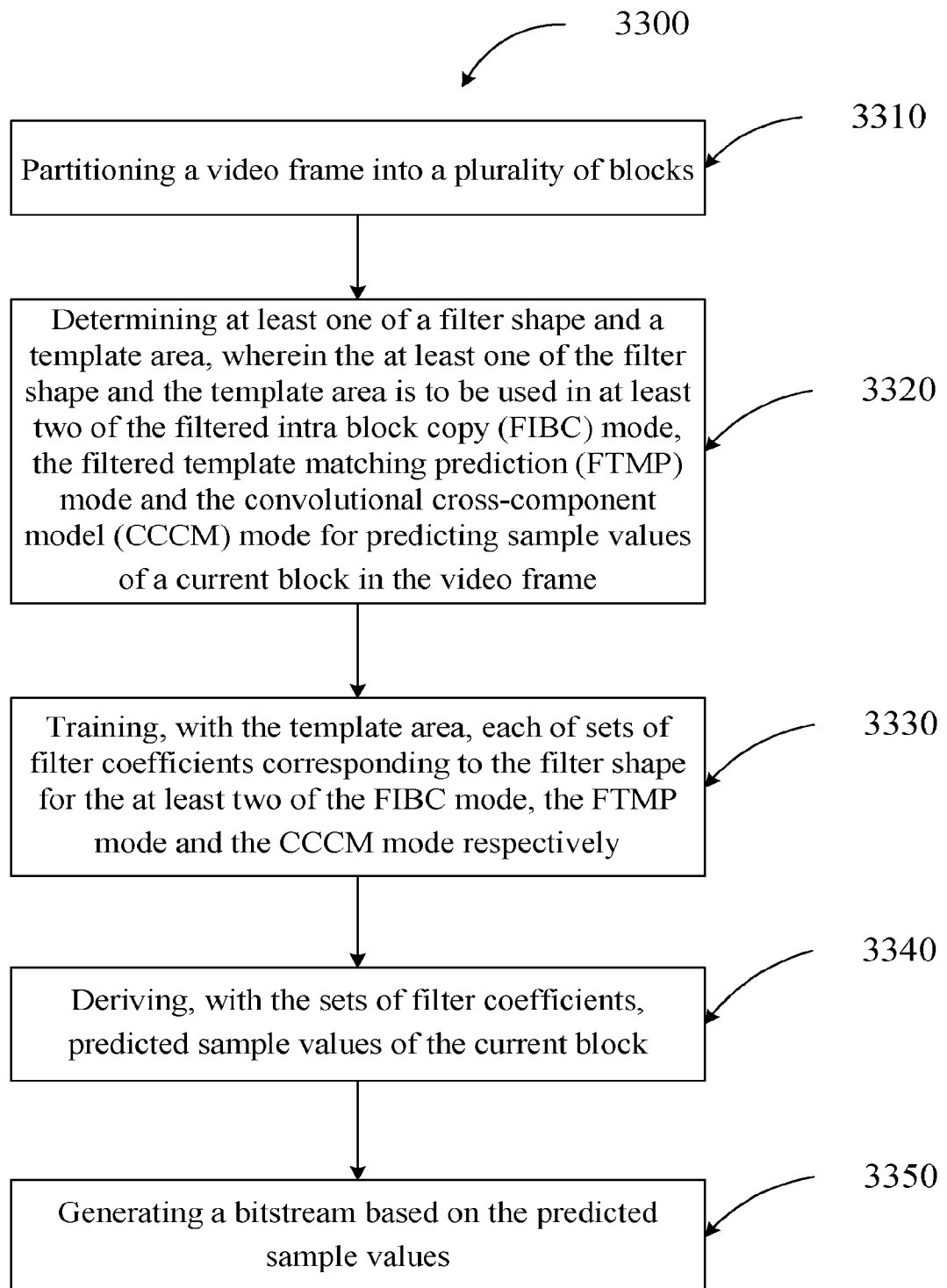


Figure 33

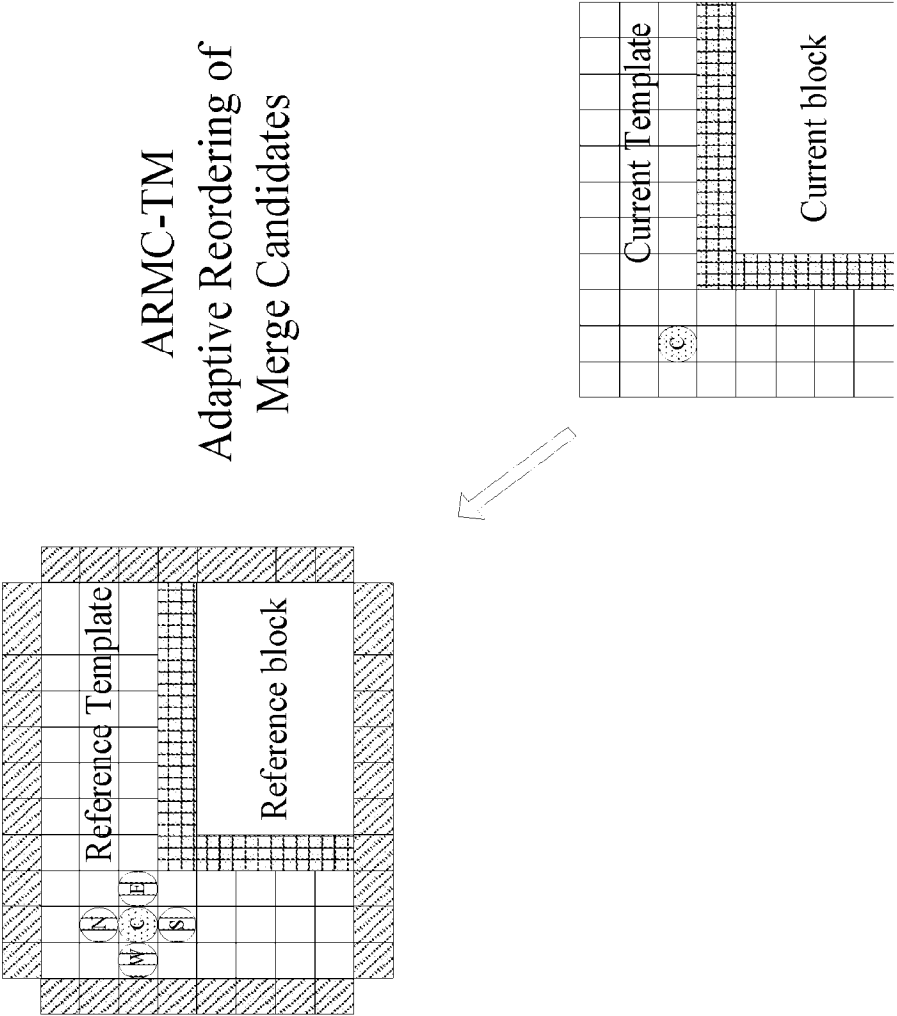


Figure 34

38/44

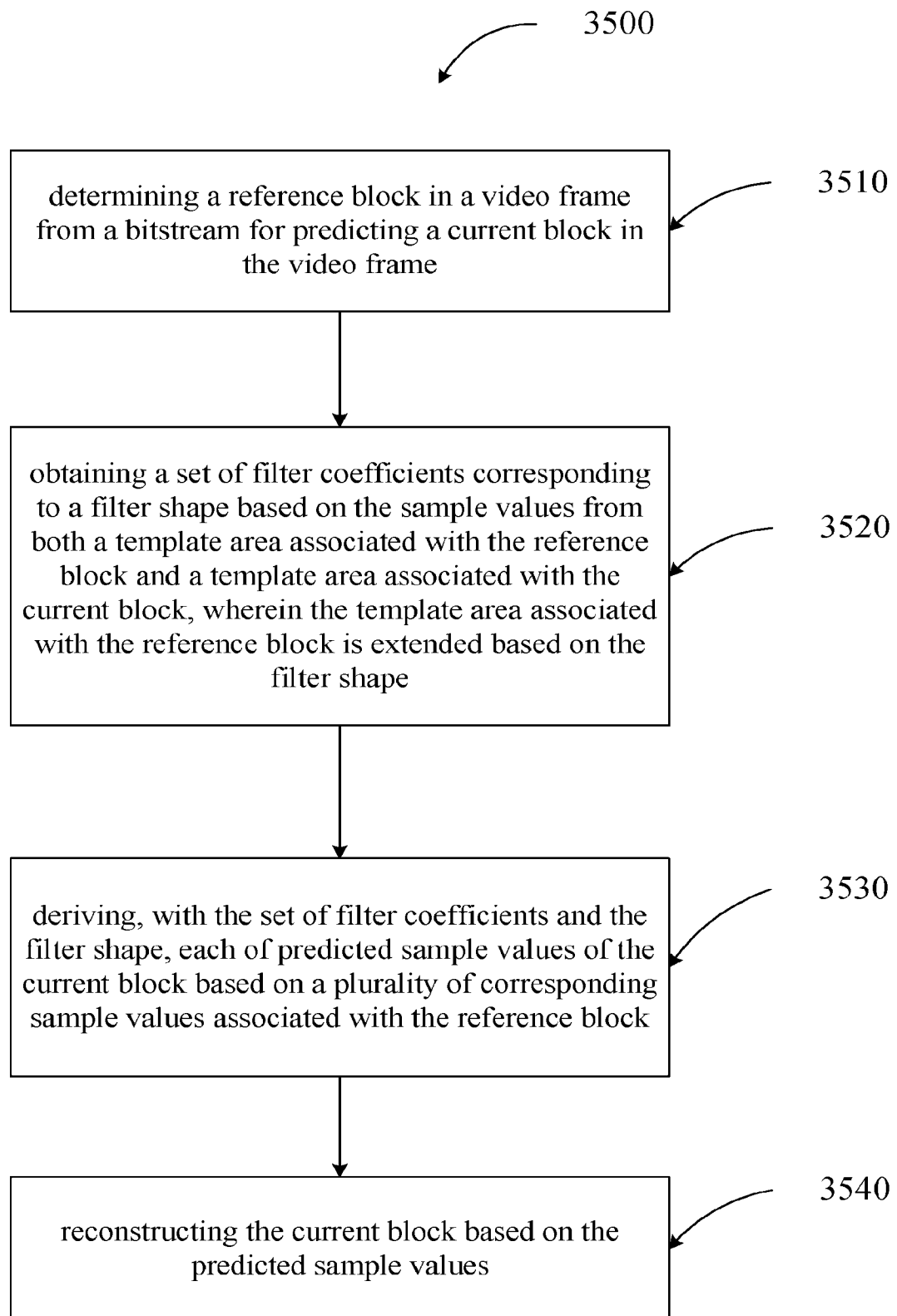


Figure 35

39/44

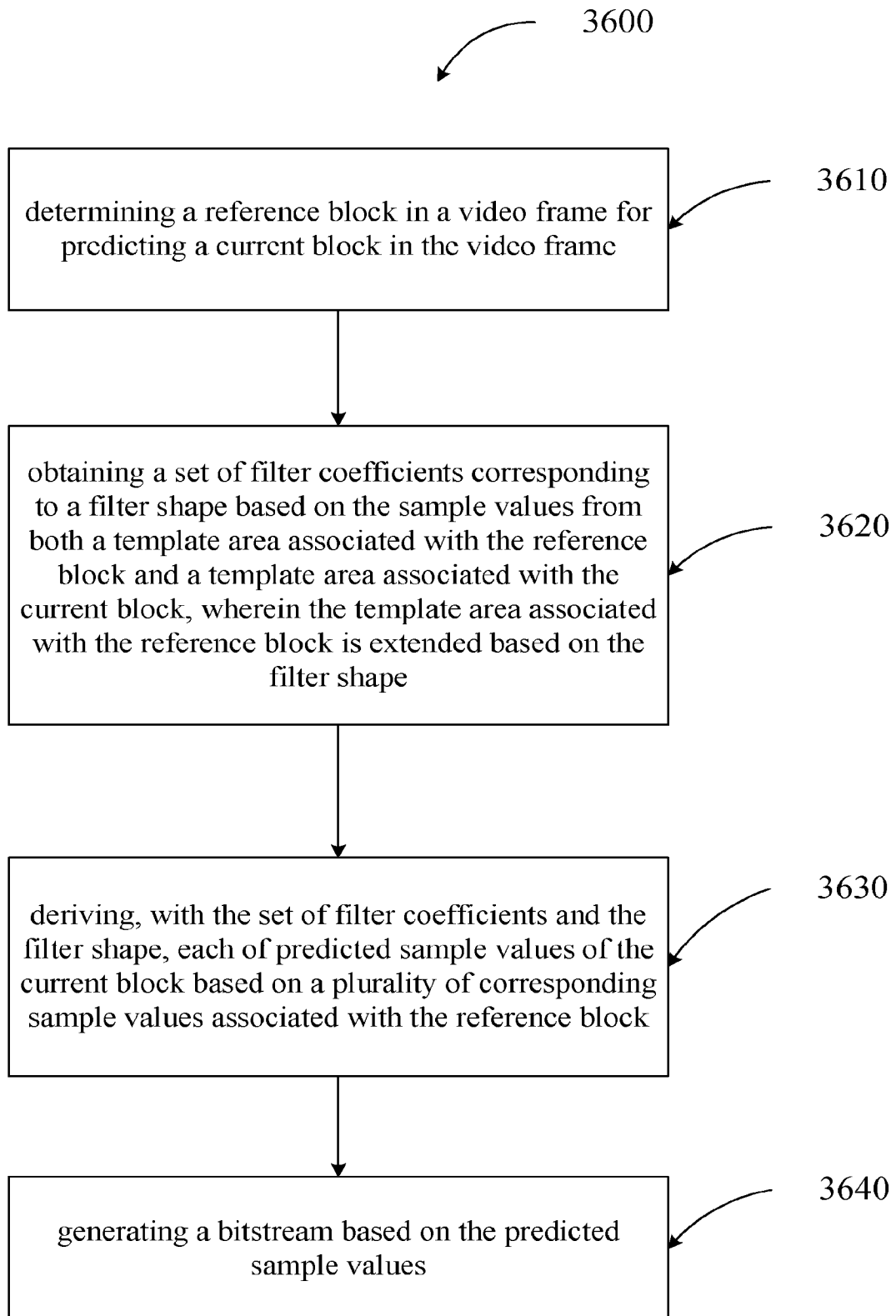


Figure 36

40/44

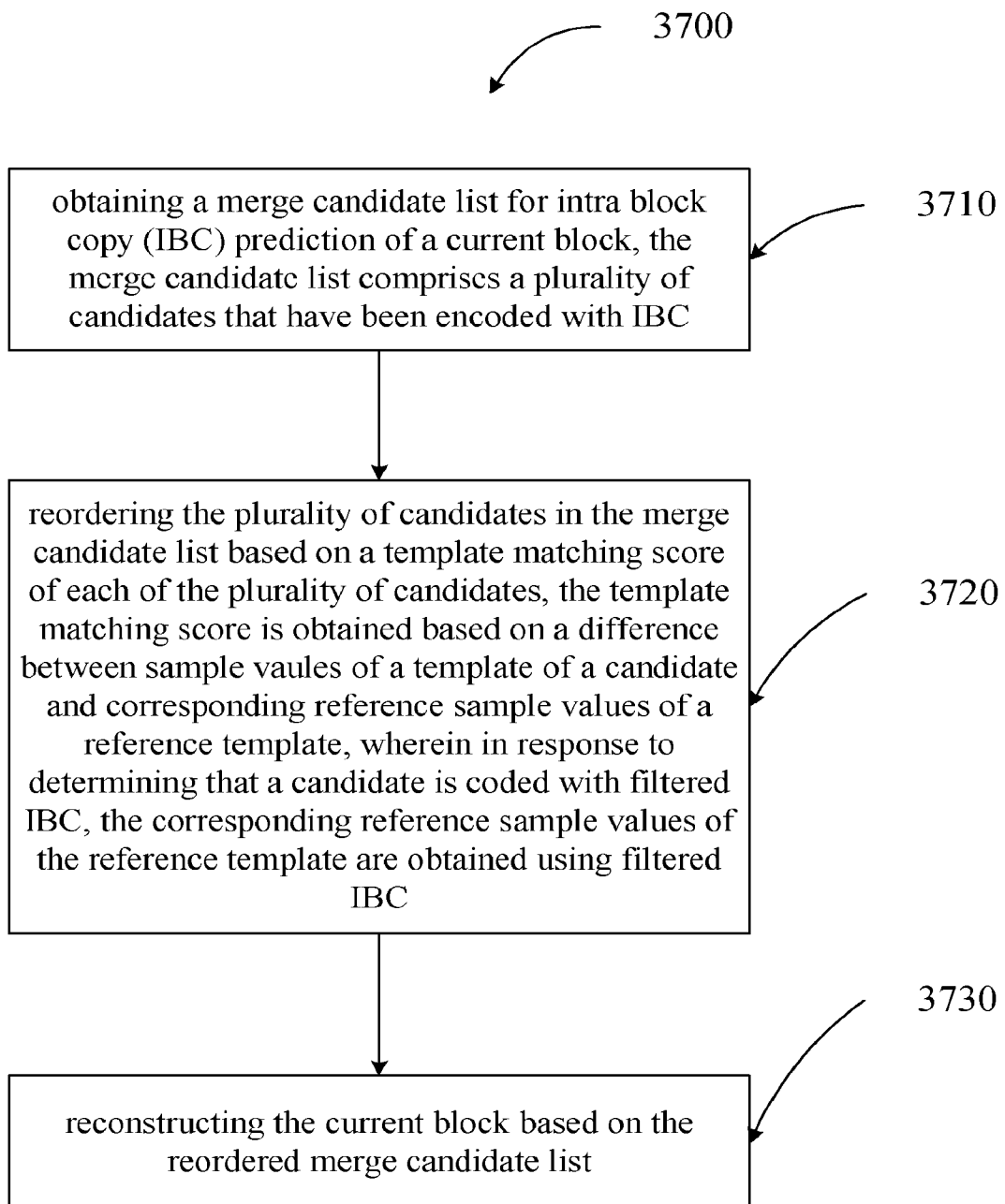


Figure 37

41/44

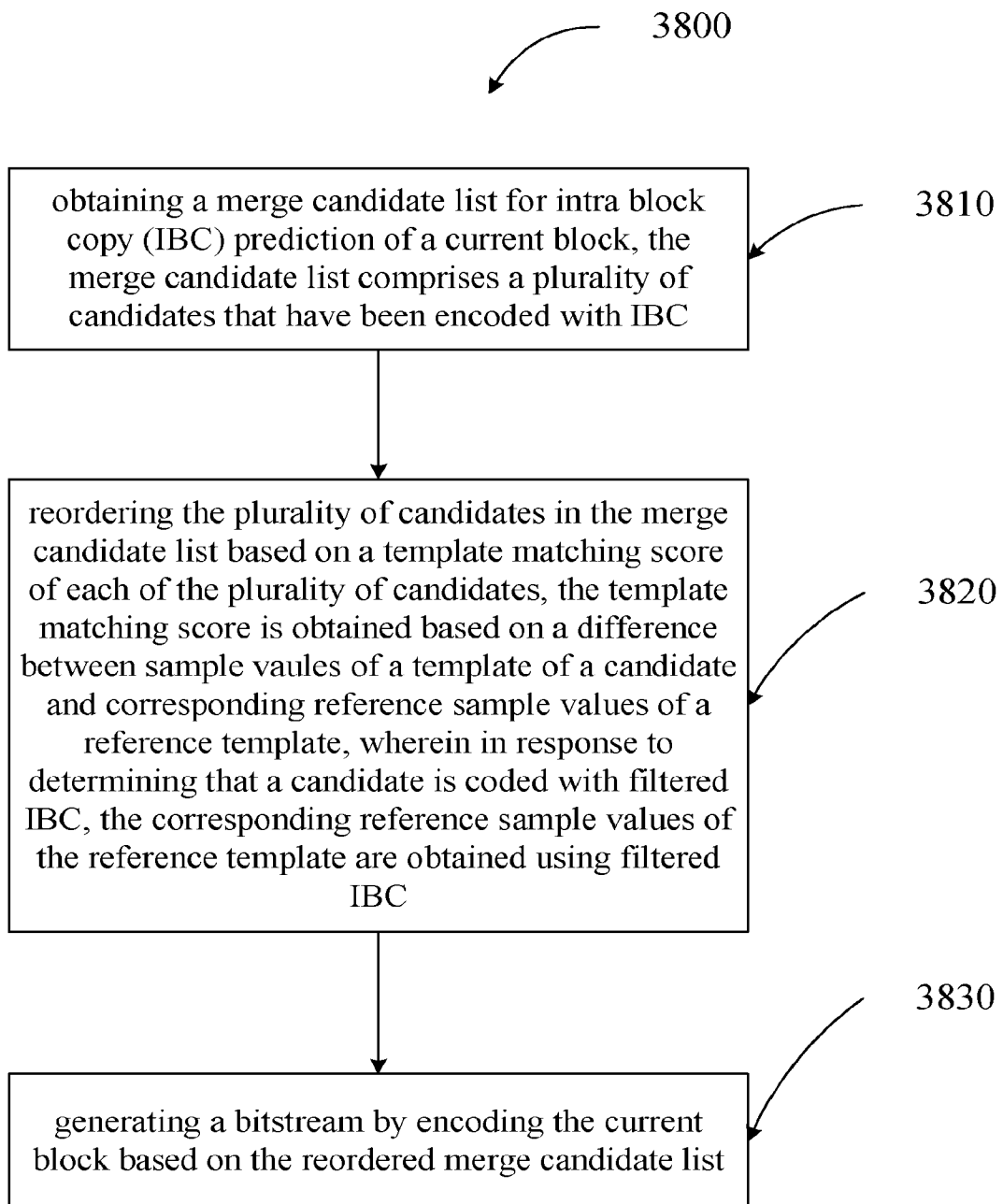


Figure 38

42/44

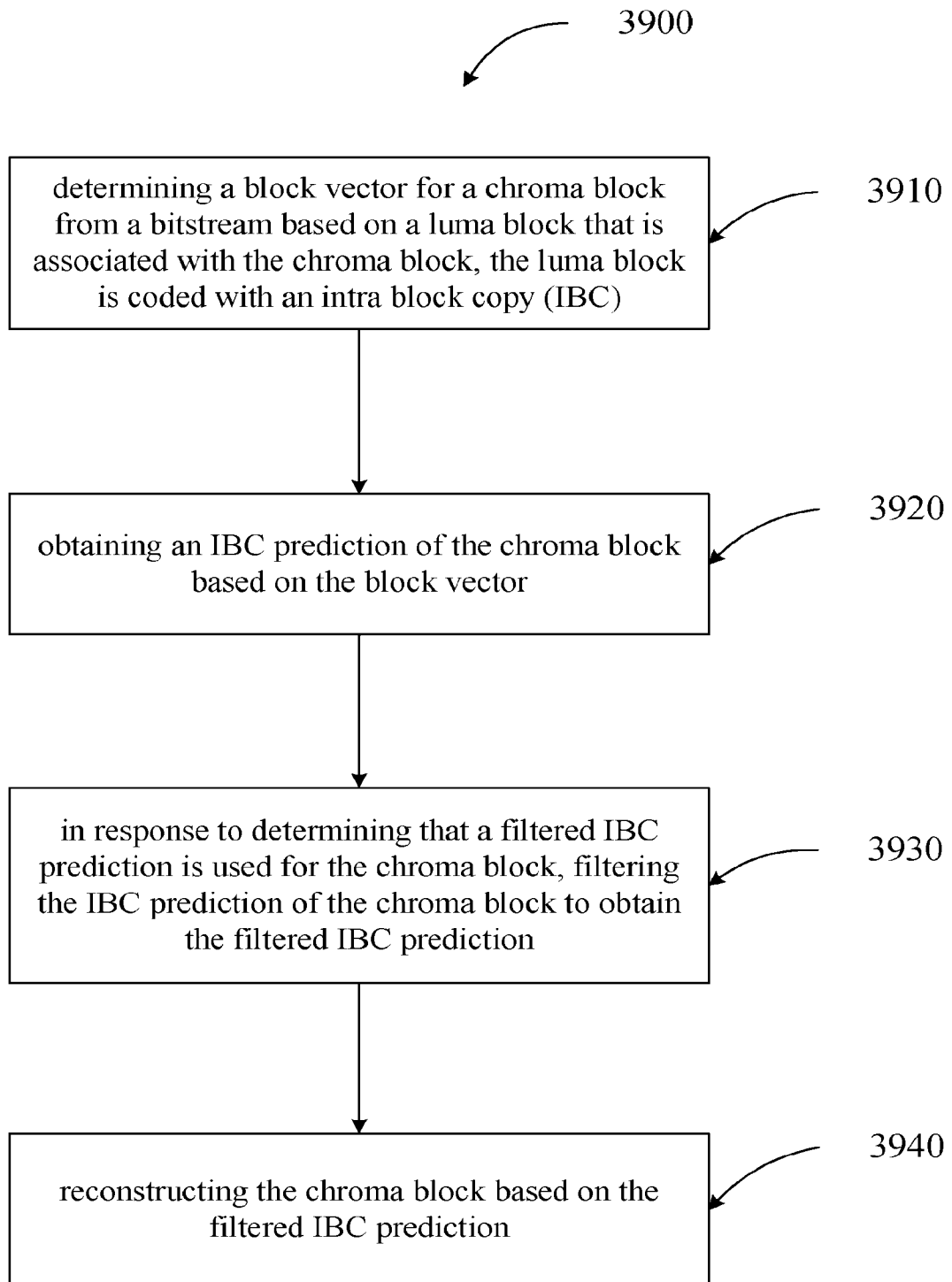


Figure 39

43/44

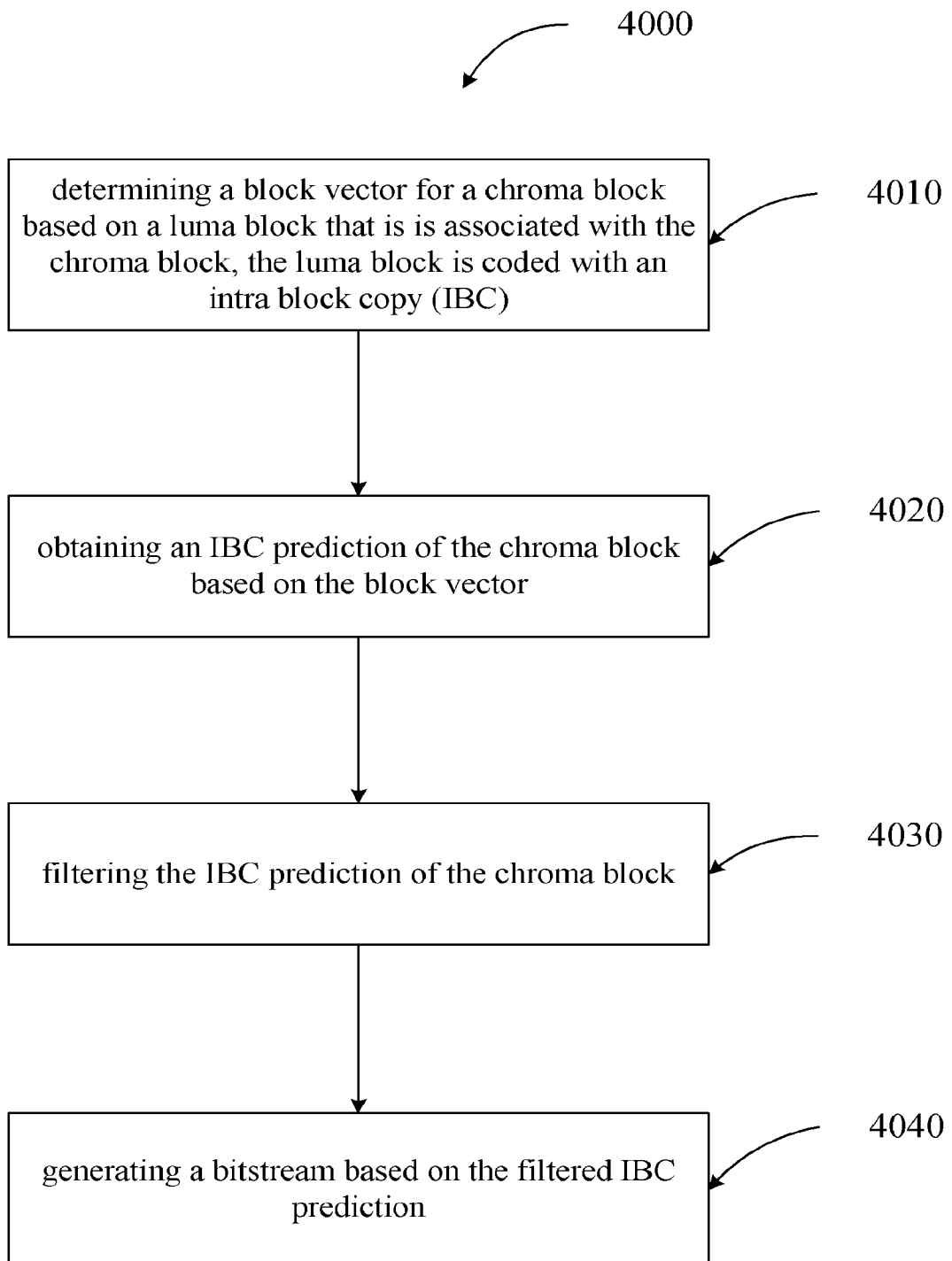


Figure 40

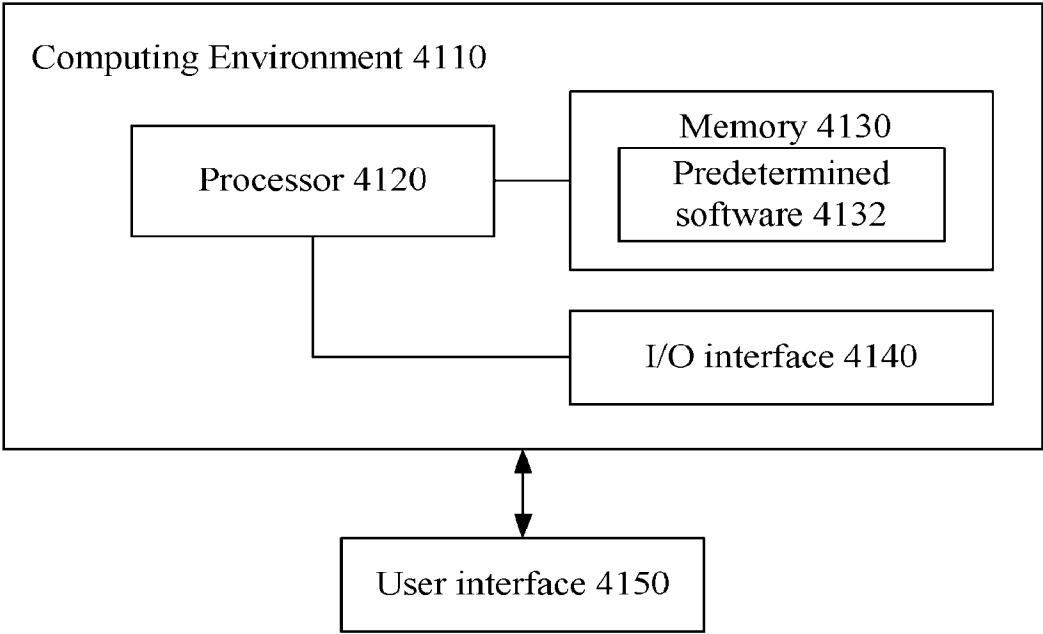


Figure 41