



## (51) International Patent Classification:

H04N 19/00 (2014.01)

## (21) International Application Number:

PCT/CN2019/094218

## (22) International Filing Date:

01 July 2019 (01.07.2019)

## (25) Filing Language:

English

## (26) Publication Language:

English

## (30) Priority Data:

201831024668 02 July 2018 (02.07.2018) IN

201831034993 17 September 2018 (17.09.2018) IN

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

(72) Inventors: **SETHURAMAN, Sriram**; Consulate-1, #1, Richmond Road, Bangalore, Karnataka, 560025 (IN). **A, Jeeva Raj**; Consulate-1, #1, Richmond Road, Bangalore, Karnataka, 560025 (IN). **KOTECCHA, Sagar**; Consulate-1, #1, Richmond Road, Bangalore, Karnataka, 560025 (IN).

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

OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

## Published:

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

## (54) Title: V REFINEMENT OF VIDEO MOTION VECTORS IN ADJACENT VIDEO DATA

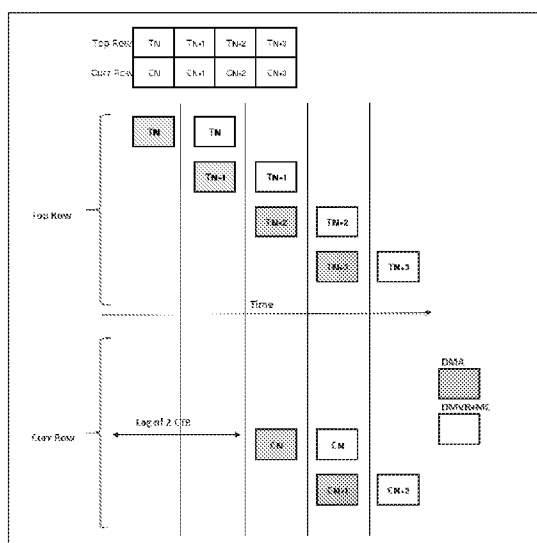


Fig. 19

(57) Abstract: A method of decoding video is provided, comprising: determining that refined motion vectors of a spatially neighboring set to be available when the refined motion vectors have been computed, wherein the availability of the refined motion vectors of the spatially neighboring set comprising at least one coding unit or sub-coding unit for a current set comprising at least one coding unit or sub coding unit; and partitioning a coding unit normatively into as many sub-coding units as the number of concurrency sets that the coding unit spans so that the data pre-fetch and decoder-side motion vector refinement and motion compensation for each sub-coding unit can happen independent of the other sub coding-units, but in a concurrent manner with other coding or sub-coding units that are part of a current concurrency set.

5

## **V REFINEMENT OF VIDEO MOTION VECTORS IN ADJACENT VIDEO DATA**

10

### **PRIORITY CLAIM AND INCORPORATION BY REFERENCE**

[0001] The current application claims the priority and benefit of Indian patent application number IN201831024668 filed on 02 July 2018 and Indian patent application number IN201831034993 filed on 17 September 2018.

[0002] Reference is made to documents JVET-K0041-v2, JVET-M0148-v2 and JVET-L0173-v2 of the Joint Video Experts Team (JVET), the contents of which are incorporated by reference, and of which the skilled person will be aware.

### **TECHNICAL FIELD**

[0003] Embodiments of the present application (disclosure) generally relate to the field of video coding, such as HEVC, and more particularly to a method for concurrent processing of Coding units within a Coded Tree Block with decoder side motion vector refinement/derivation.

### **BACKGROUND**

[0004] The amount of video data needed to depict even a relatively short video can be substantial, which may result in difficulties when the data is to be streamed or otherwise communicated across a communications network with limited bandwidth capacity. Thus, video data is generally compressed before being communicated across modern day telecommunications networks. The size of a video could also be an issue when the video is stored on a storage device because memory resources may be limited. Video compression devices often use software and/or hardware at the source to code the video data prior to transmission or storage, thereby decreasing the quantity of data needed to represent digital video images. The compressed data is then received at the destination by a video decompression device that decodes the video data. With

5        limited network resources and ever increasing demands of higher video quality, improved compression and decompression techniques that improve compression ratio with little to no sacrifice in image quality are desirable.

### SUMMARY

10        [0005] Embodiments of the present application (or the present disclosure) provide apparatuses and methods for encoding and decoding. Particular embodiments are outlined in the attached independent claims, with other embodiments in the dependent claims.

15        [0006] In a first manner, a decoding method is provided, comprising: determining that refined motion vectors of a spatially neighboring set to be available when the refined motion vectors have been computed, wherein the availability of the refined motion vectors of the spatially neighboring set comprising at least one coding unit or sub-coding unit for a current set comprising at least one coding unit or sub coding unit.

20        [0007] Further, the first method may comprise. partitioning a coding unit normatively into as many sub-coding units as the number of concurrency sets that the coding unit spans so that the data pre-fetch and decoder-side motion vector refinement and motion compensation for each sub-coding unit can happen independent of the other sub coding-units, but in a concurrent manner with other coding or sub-coding units that are part of a current concurrency set.

25        [0008] Wherein the method may be performed in a data pre-fetch stage of a current set; wherein the refined motion vectors have been computed in a pipeline stage of the current set, wherein the pipeline stage is an ahead stage of the data pre-fetch stage.

30        [0009] In a second manner, a method for using unrefined motion vectors is provided, the unrefined motion vectors being for video data of a neighbor coding unit (CU) that falls in a preceding pipeline slot (also described herein as a processing stage or cycle) to the current coding unit's concurrency set, to perform pre-fetch and use the refined motion vectors of a neighbor CU that does

5       not fall within the same concurrency set as start of refinement motion vector center. Further, using padded samples whenever the refinement requires samples that are not pre-fetched based on a configurable additional sample fetch range around the pre-fetch center.

[0010] The above manners may be combined in a decoding apparatus.

10

[0011] In a third manner, a decoding apparatus is provided, which comprises modules/units/components/circuits to perform at least a part of the steps of the method in the first manner.

15

[0012] In a fourth manner, a decoding apparatus is provided, which comprises a memory storing instructions; and a processor coupled to the memory, the processor configured to execute the instructions stored in the memory to cause the processor to perform the method of the first manner.

20

[0013] In a fifth manner, a computer-readable storage medium is provided, which having a program recorded thereon; where the program makes the computer execute method of the first manner.

25

[0014] In a sixth manner, computer program is provided, which is configured to cause a computer to execute method of the first manner.

[0015] For the purpose of clarity, any one of the foregoing embodiments may be combined with any one or more of the other foregoing embodiments to create a new embodiment within the scope of the present disclosure.

30

[0016] It will be understood that one aspect of the disclosure thus relates to an improvement in decoder side motion vector refinement. Decoder side motion vector refinement affects concurrent processing of coding units when the refined motion vector of a coding unit becomes a predictor for another coding unit or a starting point for decoder-side refinement of a coding unit, in other words when there is a dependency this can stall the pipeline.

[0017] In embodiments of the disclosure, the disclosed method determines the availability of refined motion vectors of coding units in such a way that ideally

5 all coding units within a coded tree block can configure their data pre-fetch in a concurrent manner in a given stage of regular (e.g. CTB-level) pipeline. By making one or more motion vectors of neighboring units available, the corresponding one or more motion vectors of neighboring units is already available (i.e. completed) from an earlier stage in the processing cycle. Thus the  
10 said available refined motion vectors can be used in obtaining a refined motion vector for other neighboring unit at a later stage. Embodiments of the disclosure additionally perform their refinement process in a concurrent manner in a subsequent e.g. next stage of the CTB-level pipeline. The concept of a lag between the top CTB row and current CTB row (such as adjacent) is utilized in  
15 determining the availability. Also, the concept of a concurrency set is introduced to normatively partition some coding units, if necessary, into sub-coding-units to meet the concurrency requirements of the pipeline.

[0018] Compared to not using any refined motion vector from the current picture, the approach disclosed herein provides a higher coding gain while  
20 ensuring that the dependency does not overly constrain the hardware implementation of the refinement process. In addition, by pre-fetching around an unrefined motion vector and using a normative padding process to access samples that go outside the normative amount of pre-fetched samples, the pipeline latency is further reduced to make even left or top-right CTB refined  
25 MVs to be used for refinement of current CTB CUs. The process is also extended to finer granularities than CTB level.

[0019] These and other features will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings and claims.

## 30 BRIEF DESCRIPTION OF THE DRAWINGS

[0020] For a more complete understanding of this disclosure, reference is now made to the following brief description, taken in connection with the

5 accompanying drawings and detailed description, wherein like reference numerals represent like parts.

[0021]FIG. 1A is a block diagram illustrating an example coding system that may implement embodiments of the invention.

10 [0022]FIG. 1B is a block diagram illustrating another example coding system that may implement embodiments of the invention.

[0023]FIG. 2 is a block diagram illustrating an example video encoder that may implement embodiments of the invention.

[0024]FIG. 3 is a block diagram illustrating an example of a video decoder that may implement embodiments of the invention.

15 [0025]FIG. 4 is a schematic diagram of a network device.

[0026]FIG. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from FIG. 1A according to an exemplary embodiment.

20 [0027]FIG. 6 is a graphical illustration for MCP based on a translational motion model.

[0028]FIG. 7 is a graphical illustration for an overview block diagram of the HEVC inter-picture prediction.

[0029]FIG. 8 is a graphical illustration for the relationship between spatial MVP candidates and spatially neighboring blocks.

25 [0030]FIG. 9 is a graphical illustration for the derivation process flow for the two spatial candidates A and B.

[0031]FIG. 10 is a graphical illustration for CTU partitioning.

[0032]FIG. 11 is a graphical illustration for sub-CUs.

30 [0033]FIG. 12 is a graphical illustration for the relationship between CU and sub-CUs.

[0034]FIG. 13 is a graphical illustration for the bilateral matching to derive motion information of the current CU.

5       [0035]FIG. 14 is a graphical illustration for template matching to derive motion information of the current CU.

      [0036]FIG. 15 is a graphical illustration for the motion associated to the block passing through a 4×4 block.

      [0037]FIG. 16 is a graphical illustration for MV0' and MV1'.

10       [0038]FIG. 17 is a graphical illustration for a regular pipeline at CTB level for lag0.

      [0039]FIG. 18 is a graphical illustration for the relationship between top row and current row.

15       [0040]FIG. 19 is a graphical illustration for another relationship between top row and current row.

      [0041]FIG. 20a is a graphical illustration for another relationship between top row and current row.

      [0042]FIG. 20b is an enlargement of the top half of FIG 20a.

      [0043]FIG. 20c is an enlargement of the top half of FIG 20a.

20       [0044]FIG. 21 is a graphical illustration for a regular pipeline at CTB level for lag 2.

      [0045]FIG. 22 is a graphical illustration for pattern.

      [0046]FIG. 23 is a graphical illustration for ErrorSurface.

25                                   **DETAILED DESCRIPTION**

      [0047]It should be understood at the outset that although an illustrative implementation of one or more embodiments are provided below, the disclosed systems and/or methods may be implemented using any number of techniques, whether currently known or in existence. The disclosure should in no way be  
30       limited to the illustrative implementations, drawings, and techniques illustrated below, including the exemplary designs and implementations illustrated and described herein, but may be modified within the scope of the appended claims along with their full scope of equivalents.

5       [0048]FIG. 1A is a block diagram illustrating an example coding system 10 that may utilize bidirectional prediction techniques. As shown in FIG. 1A, the coding system 10 includes a source device 12 that provides encoded video data to be decoded at a later time by a destination device 14. In particular, the source device 12 may provide the video data to destination device 14 via a computer-  
10       readable medium 16. Source device 12 and destination device 14 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as so-called “smart” phones, so-called “smart” pads, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device,  
15       or the like. In some cases, source device 12 and destination device 14 may be equipped for wireless communication.

      [0049]Destination device 14 may receive the encoded video data to be decoded via computer-readable medium 16. Computer-readable medium 16 may comprise any type of medium or device capable of moving the encoded video  
20       data from source device 12 to destination device 14. In one example, computer-readable medium 16 may comprise a communication medium to enable source device 12 to transmit encoded video data directly to 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  
25       transmitted to 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  
30       communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 12 to destination device 14.

5        [0050] In some examples, encoded data may be output from output interface 22 to a storage device. Similarly, encoded data may be accessed from the storage device by input interface. The storage device may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, digital video disks (DVD)s, Compact Disc Read-Only Memories (CD-ROMs), flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data. In a further example, the storage device may correspond to a file server or another intermediate storage device that may store the encoded video generated by source device 12. Destination device 14 may access stored video data from the storage device via streaming or download. The file server may be any type of server capable of storing encoded video data and transmitting that encoded video data to the destination device 14. Example 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. Destination device 14 may access the encoded video data through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a 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 encoded video data from the storage device may be a streaming transmission, a download transmission, or a combination thereof.

10        [0051] The techniques of this disclosure are not necessarily limited to wireless applications or settings. The techniques may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications. In some examples, coding system 10 may be configured to support

15       

20       

25       

30

5 one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

[0052] In the example of FIG. 1A, source device 12 includes video source 18, video encoder 20, and output interface 22. Destination device 14 includes input interface 28, video decoder 30, and display device 32. In accordance with this  
10 disclosure, video encoder 20 of source device 12 and/or the video decoder 30 of the destination device 14 may be configured to apply the techniques for bidirectional prediction. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 12 may receive video data from an external video source, such as an  
15 external camera. Likewise, destination device 14 may interface with an external display device, rather than including an integrated display device.

[0053] The illustrated coding system 10 of FIG. 1A is merely one example. Techniques for bidirectional prediction may be performed by any digital video encoding and/or decoding device. Although the techniques of this disclosure  
20 generally are performed by a video coding device, the techniques may also be performed by a video encoder/decoder, typically referred to as a “CODEC.” Moreover, the techniques of this disclosure may also be performed by a video preprocessor. The video encoder and/or the decoder may be a graphics processing unit (GPU) or a similar device.

[0054] Source device 12 and destination device 14 are merely examples of such  
25 coding devices in which source device 12 generates coded video data for transmission to destination device 14. In some examples, source device 12 and destination device 14 may operate in a substantially symmetrical manner such that each of the source and destination devices 12, 14 includes video encoding  
30 and decoding components. Hence, coding system 10 may support one-way or two-way video transmission between video devices 12, 14, e.g., for video streaming, video playback, video broadcasting, or video telephony.

5       [0055] Video source 18 of source device 12 may include a video capture device, such as a video camera, a video archive containing previously captured video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 18 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and  
10       computer-generated video.

      [0056] In some cases, when video source 18 is a video camera, source device 12 and destination device 14 may form so-called camera phones or video phones. As mentioned above, however, the techniques described in this disclosure may be applicable to video coding in general, and may be applied to wireless and/or  
15       wired applications. In each case, the captured, pre-captured, or computer-generated video may be encoded by video encoder 20. The encoded video information may then be output by output interface 22 onto a computer-readable medium 16.

      [0057] Computer-readable medium 16 may include transient media, such as a  
20       wireless broadcast or wired network transmission, or storage media (that is, non-transitory storage media), such as a hard disk, flash drive, compact disc, digital video disc, Blu-ray disc, or other computer-readable media. In some examples, a network server (not shown) may receive encoded video data from source device 12 and provide the encoded video data to destination device 14, e.g., via  
25       network transmission. Similarly, a computing device of a medium production facility, such as a disc stamping facility, may receive encoded video data from source device 12 and produce a disc containing the encoded video data. Therefore, computer-readable medium 16 may be understood to include one or more computer-readable media of various forms, in various examples.

30       [0058] Input interface 28 of destination device 14 receives information from computer-readable medium 16. The information of computer-readable medium 16 may include syntax information defined by video encoder 20, which is also used by video decoder 30, that includes syntax elements that describe

5 characteristics and/or processing of blocks and other coded units, e.g., group of pictures (GOPs). Display device 32 displays the decoded video data to a user, and may comprise any of a variety of display devices such as a cathode ray tube (CRT), a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

10 [0059] Video encoder 20 and video decoder 30 may operate according to a video coding standard, such as the High Efficiency Video Coding (HEVC) standard presently under development, and may conform to the HEVC Test Model (HM). Alternatively, video encoder 20 and video decoder 30 may operate according to other proprietary or industry standards, such as the International

15 Telecommunications Union Telecommunication Standardization Sector (ITU-T) H.264 standard, alternatively referred to as Motion Picture Expert Group (MPEG)-4, Part 10, Advanced Video Coding (AVC), H.265/HEVC, or extensions of such standards. The techniques of this disclosure, however, are not limited to any particular coding standard. Other examples of video coding

20 standards include MPEG-2 and ITU-T H.263. Although not shown in FIG. 1A, in some aspects, video encoder 20 and video decoder 30 may each be integrated with an audio encoder and decoder, and may include appropriate multiplexer-demultiplexer (MUX-DEMUX) units, or other hardware and software, to handle encoding of both audio and video in a common data stream or separate data

25 streams. If applicable, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

[0060] It will thus be understood that terms used in this disclosure may, but not necessarily, have the particular technical meaning and/or definition provided in

30 the HEVC standard. For example, in the HEVC standard, a Coding tree unit (CTU) is the basic processing unit, which conceptually corresponds in structure to macroblock units that were used in several previous video standards.

5 HEVC initially divides the picture into CTUs which are then divided for each luma/chroma component into coding tree blocks (CTBs). A CTB can be  $64 \times 64$ ,  $32 \times 32$ , or  $16 \times 16$  with a larger pixel block size usually increasing the coding efficiency. CTBs are then divided into same size or smaller one or more coding units (CUs), so that the CTU size is also the largest coding unit size. The  
10 arrangement of CUs in a CTB is known as a quadtree since a subdivision results in four smaller regions. CUs are then divided into prediction units (PUs) and / or transform units (Tus) of either intra-picture or inter-picture prediction type which can vary in size from  $64 \times 64$  to  $4 \times 4$ .

[0061] Video encoder 20 and video decoder 30 each may be implemented as any  
15 of a variety of suitable encoder 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 the techniques are implemented partially in software, a device may store instructions for the software in a  
20 suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 20 and video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including  
25 video encoder 20 and/or video decoder 30 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0062] **Fig. 1B** is an illustrative diagram of an example video coding system 40  
30 including encoder 20 of fig. 2 and/or decoder 30 of fig. 3 according to an exemplary embodiment. The system 40 can implement techniques of this present application. In the illustrated implementation, video coding system 40 may

5 include imaging device(s) 41, video encoder 20, video decoder 30 (and/or a video coder implemented via logic circuitry 54 of processing unit(s) 46), an antenna 42, one or more processor(s) 43, one or more memory store(s) 44, and/or a display device 45.

[0063] As illustrated, imaging device(s) 41, antenna 42, processing unit(s) 46,  
10 logic circuitry 54, video encoder 20, video decoder 30, processor(s) 43, memory store(s) 44, and/or display device 45 may be capable of communication with one another. As discussed, although illustrated with both video encoder 20 and video decoder 30, video coding system 40 may include only video encoder 20 or only video decoder 30 in various examples.

15 [0064] As shown, in some examples, video coding system 40 may include antenna 42. Antenna 42 may be configured to transmit or receive an encoded bitstream of video data, for example. Further, in some examples, video coding system 40 may include display device 45. Display device 45 may be configured to present video data. As shown, in some examples, logic circuitry 54 may be  
20 implemented via processing unit(s) 46. Processing unit(s) 46 may include application-specific integrated circuit (ASIC) logic, graphics processor(s), general purpose processor(s), or the like. Video coding system 40 also may include optional processor(s) 43, which may similarly include application-specific integrated circuit (ASIC) logic, graphics processor(s), general purpose  
25 processor(s), or the like. In some examples, logic circuitry 54 may be implemented via hardware, video coding dedicated hardware, or the like, and processor(s) 43 may implemented general purpose software, operating systems, or the like. In addition, memory store(s) 44 may be any type of memory such as volatile memory (e.g., Static Random Access Memory (SRAM), Dynamic  
30 Random Access Memory (DRAM), etc.) or non-volatile memory (e.g., flash memory, etc.), and so forth. In a non-limiting example, memory store(s) 44 may be implemented by cache memory. In some examples, logic circuitry 54 may access memory store(s) 44 (for implementation of an image buffer for example).

5 In other examples, logic circuitry 47 and/or processing unit(s) 46 may include memory stores (e.g., cache or the like) for the implementation of an image buffer or the like.

[0065] In some examples, video encoder 20 implemented via logic circuitry may include an image buffer (e.g., via either processing unit(s) 46 or memory store(s) 10 44)) and a graphics processing unit (e.g., via processing unit(s) 46). The graphics processing unit may be communicatively coupled to the image buffer. The graphics processing unit may include video encoder 20 as implemented via logic circuitry 47 to embody the various modules as discussed with respect to FIG. 2 and/or any other encoder system or subsystem described herein. The logic 15 circuitry may be configured to perform the various operations as discussed herein.

[0066] Video decoder 30 may be implemented in a similar manner as implemented via logic circuitry 47 to embody the various modules as discussed with respect to decoder 30 of FIG. 3 and/or any other decoder system or 20 subsystem described herein. In some examples, video decoder 30 may be implemented via logic circuitry may include an image buffer (e.g., via either processing unit(s) 420 or memory store(s) 44)) and a graphics processing unit (e.g., via processing unit(s) 46). The graphics processing unit may be communicatively coupled to the image buffer. The graphics processing unit may 25 include video decoder 30 as implemented via logic circuitry 47 to embody the various modules as discussed with respect to FIG. 3 and/or any other decoder system or subsystem described herein.

[0067] In some examples, antenna 42 of video coding system 40 may be configured to receive an encoded bitstream of video data. As discussed, the 30 encoded bitstream may include data, indicators, index values, mode selection data, or the like associated with encoding a video frame as discussed herein, such as data associated with the coding partition (e.g., transform coefficients or quantized transform coefficients, optional indicators (as discussed), and/or data

5 defining the coding partition). Video coding system 40 may also include video decoder 30 coupled to antenna 42 and configured to decode the encoded bitstream. The display device 45 configured to present video frames.

[0068] FIG. 2 is a block diagram illustrating an example of video encoder 20 that  
10 may implement the techniques of the present application. Video encoder 20 may perform intra- and inter-coding of video blocks within video slices. Intra-coding relies on spatial prediction to reduce or remove spatial redundancy in video within a given video frame or picture. Inter-coding relies on temporal prediction to reduce or remove temporal redundancy in video within adjacent frames or  
15 pictures of a video sequence. Intra-mode (I mode) may refer to any of several spatial based coding modes. Inter-modes, such as uni-directional prediction (P mode) or bi-prediction (B mode), may refer to any of several temporal-based coding modes.

[0069] As shown in FIG. 2, video encoder 20 receives a current video block  
20 within a video frame to be encoded. In the example of FIG. 2, video encoder 20 includes mode select unit 40, reference frame memory 64, summer 50, transform processing unit 52, quantization unit 54, and entropy coding unit 56. Mode select unit 40, in turn, includes motion compensation unit 44, motion estimation unit 42, intra-prediction unit 46, and partition unit 48. For video block  
25 reconstruction, video encoder 20 also includes inverse quantization unit 58, inverse transform unit 60, and summer 62. A deblocking filter (not shown in FIG. 2) may also be included to filter block boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of summer 62. Additional filters (in loop or post loop)  
30 may also be used in addition to the deblocking filter. Such filters are not shown for brevity, but if desired, may filter the output of summer 50 (as an in-loop filter).

5        [0070] During the encoding process, video encoder 20 receives a video frame or slice to be coded. The frame or slice may be divided into multiple video blocks. Motion estimation unit 42 and motion compensation unit 44 perform inter-predictive coding of the received video block relative to one or more blocks in one or more reference frames to provide temporal prediction. Intra-prediction  
10        unit 46 may alternatively perform intra-predictive coding of the received video block relative to one or more neighboring blocks in the same frame or slice as the block to be coded to provide spatial prediction. Video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

15        [0071] Moreover, partition unit 48 may partition blocks of video data into sub-blocks, based on evaluation of previous partitioning schemes in previous coding passes. For example, partition unit 48 may initially partition a frame or slice into largest coding units (LCUs), and partition each of the LCUs into sub-coding  
20        units (sub-CUs) based on rate-distortion analysis (e.g., rate-distortion optimization). Mode select unit 40 may further produce a quadtree data structure indicative of partitioning of a LCU into sub-CUs. Leaf-node CUs of the quadtree may include one or more prediction units (PUs) and one or more transform units (TUs).

25        [0072] The present disclosure uses the term “block” to refer to any of a CU, PU, or TU, in the context of HEVC, or similar data structures in the context of other standards (e.g., macroblocks and sub-blocks thereof in H.264/AVC). A CU includes a coding node, PUs, and TUs associated with the coding node. A size  
30        of the CU corresponds to a size of the coding node and is square in shape. The size of the CU may range from 8×8 pixels up to the size of the treeblock with a maximum of 64×64 pixels or greater. Each CU may contain one or more PUs and one or more TUs. Syntax data associated with a CU may describe, for example, partitioning of the CU into one or more PUs. Partitioning modes may differ between whether the CU is skip or direct mode encoded, intra-prediction

5 mode encoded, or inter-prediction mode encoded. PUs may be partitioned to be non-square in shape. Syntax data associated with a CU may also describe, for example, partitioning of the CU into one or more TUs according to a quadtree. In an embodiment, a CU, PU, or TU can be square or non-square (e.g., rectangular) in shape.

10 [0073] Mode select unit 40 may select one of the coding modes, intra or inter, e.g., based on error results, and provides the resulting intra- or inter-coded block to summer 50 to generate residual block data and to summer 62 to reconstruct the encoded block for use as a reference frame. Mode select unit 40 also provides syntax elements, such as motion vectors, intra-mode indicators,

15 partition information, and other such syntax information, to entropy coding unit 56.

[0074] Motion estimation unit 42 and motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation unit 42, is the process of generating

20 motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a PU of a video block within a current video frame or picture relative to a predictive block within a reference frame (or other coded unit) relative to the current block being coded within the current frame (or other coded unit). A predictive block is a block that is found to

25 closely match the 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 examples, video encoder 20 may calculate values for sub-integer pixel positions of reference pictures stored in reference frame memory 64. For example, video encoder 20 may interpolate

30 values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation unit 42 may perform a motion search relative to the full pixel positions and

5 fractional pixel positions and output a motion vector with fractional pixel precision.

[0075] Motion estimation unit 42 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a predictive block of a reference picture. The reference picture may be  
10 selected from a first reference picture list (List 0) or a second reference picture list (List 1), each of which identify one or more reference pictures stored in reference frame memory 64. Motion estimation unit 42 sends the calculated motion vector to entropy encoding unit 56 and motion compensation unit 44.

[0076] Motion compensation, performed by motion compensation unit 44, may  
15 involve fetching or generating the predictive block based on the motion vector determined by motion estimation unit 42. Again, motion estimation unit 42 and motion compensation unit 44 may be functionally integrated, in some examples. Upon receiving the motion vector for the PU of the current video block, motion compensation unit 44 may locate the predictive block to which the motion vector  
20 points in one of the reference picture lists. Summer 50 forms 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, as discussed below. In general, motion estimation unit 42 performs motion estimation relative to luma components, and motion compensation unit 44 uses  
25 motion vectors calculated based on the luma components for both chroma components and luma components. Mode select unit 40 may also generate syntax elements associated with the video blocks and the video slice for use by video decoder 30 in decoding the video blocks of the video slice.

[0077] Intra-prediction unit 46 may intra-predict a current block, as an alternative  
30 to the inter-prediction performed by motion estimation unit 42 and motion compensation unit 44, as described above. In particular, intra-prediction unit 46 may determine an intra-prediction mode to use to encode a current block. In some examples, intra-prediction unit 46 may encode a current block using

5 various intra-prediction modes, e.g., during separate encoding passes, and intra-prediction unit 46 (or mode select unit 40, in some examples) may select an appropriate intra-prediction mode to use from the tested modes.

[0078] For example, intra-prediction unit 46 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and  
10 select the intra-prediction mode having the best rate-distortion characteristics among the tested modes. 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 (that is, a number of bits) used to produce the encoded block. Intra-  
15 prediction unit 46 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.

[0079] In addition, intra-prediction unit 46 may be configured to code depth blocks of a depth map using a depth modeling mode (DMM). Mode select unit  
20 40 may determine whether an available DMM mode produces better coding results than an intra-prediction mode and the other DMM modes, e.g., using rate-distortion optimization (RDO). Data for a texture image corresponding to a depth map may be stored in reference frame memory 64. Motion estimation unit 42 and motion compensation unit 44 may also be configured to inter-predict  
25 depth blocks of a depth map.

[0080] After selecting an intra-prediction mode for a block (e.g., a conventional intra-prediction mode or one of the DMM modes), intra-prediction unit 46 may provide information indicative of the selected intra-prediction mode for the block to entropy coding unit 56. Entropy coding unit 56 may encode the  
30 information indicating the selected intra-prediction mode. Video encoder 20 may include in the transmitted bitstream configuration data, which may include a plurality of intra-prediction mode index tables and a plurality of modified intra-prediction mode index tables (also referred to as codeword mapping tables),

5 definitions of encoding contexts for various blocks, and indications of a most probable intra-prediction mode, an intra-prediction mode index table, and a modified intra-prediction mode index table to use for each of the contexts.

[0081] Video encoder 20 forms a residual video block by subtracting the prediction data from mode select unit 40 from the original video block being  
10 coded. Summer 50 represents the component or components that perform this subtraction operation.

[0082] Transform processing unit 52 applies a transform, such as a discrete cosine transform (DCT) or a conceptually similar transform, to the residual block, producing a video block comprising residual transform coefficient values.  
15 Transform processing unit 52 may perform other transforms which are conceptually similar to DCT. Wavelet transforms, integer transforms, sub-band transforms or other types of transforms could also be used.

[0083] Transform processing unit 52 applies the transform to the residual block, producing a block of residual transform coefficients. The transform may convert  
20 the residual information from a pixel value domain to a transform domain, such as a frequency domain. Transform processing unit 52 may send the resulting transform coefficients to quantization unit 54. Quantization unit 54 quantizes the transform coefficients to further reduce bit rate. The quantization process may reduce the bit depth associated with some or all of the coefficients. The  
25 degree of quantization may be modified by adjusting a quantization parameter. In some examples, quantization unit 54 may then perform a scan of the matrix including the quantized transform coefficients. Alternatively, entropy encoding unit 56 may perform the scan.

[0084] Following quantization, entropy coding unit 56 entropy codes the quantized transform coefficients. For example, entropy coding unit 56 may  
30 perform 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)

5 coding or another entropy coding technique. In the case of context-based entropy coding, context may be based on neighboring blocks. Following the entropy coding by entropy coding unit 56, the encoded bitstream may be transmitted to another device (e.g., video decoder 30) or archived for later transmission or retrieval.

10 [0085]Inverse quantization unit 58 and inverse transform unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual block in the pixel domain, e.g., for later use as a reference block. Motion compensation unit 44 may calculate a reference block by adding the residual block to a predictive block of one of the frames of reference frame memory 64.

15 Motion compensation unit 44 may also apply one or more interpolation filters to the reconstructed residual block to calculate sub-integer pixel values for use in motion estimation. Summer 62 adds the reconstructed residual block to the motion compensated prediction block produced by motion compensation unit 44 to produce a reconstructed video block for storage in reference frame memory

20 64. The reconstructed video block may be used by motion estimation unit 42 and motion compensation unit 44 as a reference block to inter-code a block in a subsequent video frame.

[0086]Other structural variations of the video encoder 20 can be used to encode the video stream. For example, a non-transform based encoder 20 can quantize

25 the residual signal directly without the transform processing unit 52 for certain blocks or frames. In another implementation, an encoder 20 can have the quantization unit 54 and the inverse quantization unit 58 combined into a single unit.

[0087]FIG. 3 is a block diagram illustrating an example of video decoder 30 that

30 may implement the techniques of this present application. In the example of FIG. 3, video decoder 30 includes an entropy decoding unit 70, motion compensation unit 72, intra-prediction unit 74, inverse quantization unit 76, inverse transformation unit 78, reference frame memory 82, and summer 80.

5 Video decoder 30 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 20 (FIG. 2). Motion compensation unit 72 may generate prediction data based on motion vectors received from entropy decoding unit 70, while intra-prediction unit 74 may generate prediction data based on intra-prediction mode indicators received  
10 from entropy decoding unit 70.

[0088] During the decoding process, video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video slice and associated syntax elements from video encoder 20. Entropy decoding unit 70 of video decoder 30 entropy decodes the bitstream to generate quantized coefficients,  
15 motion vectors or intra-prediction mode indicators, and other syntax elements. Entropy decoding unit 70 forwards the motion vectors and other syntax elements to motion compensation unit 72. Video decoder 30 may receive the syntax elements at the video slice level and/or the video block level.

[0089] When the video slice is coded as an intra-coded (I) slice, intra prediction unit 74 may generate prediction data for a video block of the current video slice  
20 based on a signaled intra prediction mode and data from previously decoded blocks of the current frame or picture. When the video frame is coded as an inter-coded (i.e., B, P, or GPB) slice, motion compensation unit 72 produces predictive blocks for a video block of the current video slice based on the motion  
25 vectors and other syntax elements received from entropy decoding unit 70. The predictive blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference pictures stored in reference frame memory 82.

30 [0090] Motion compensation unit 72 determines prediction information for a video block of the current video slice by parsing the motion vectors and other syntax elements, and uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, motion

5 compensation unit 72 uses some of the received syntax elements to determine a prediction mode (e.g., intra- or inter-prediction) used to code the video blocks of the video slice, an inter-prediction slice type (e.g., B slice, P slice, or GPB slice), construction information for one or more of the reference picture lists for the slice, motion vectors for each inter-encoded video block of the slice, inter-  
10 prediction status for each inter-coded video block of the slice, and other information to decode the video blocks in the current video slice.

[0091] Motion compensation unit 72 may also perform interpolation based on interpolation filters. Motion compensation unit 72 may use interpolation filters as used by video encoder 20 during encoding of the video blocks to calculate  
15 interpolated values for sub-integer pixels of reference blocks. In this case, motion compensation unit 72 may determine the interpolation filters used by video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[0092] Data for a texture image corresponding to a depth map may be stored in  
20 reference frame memory 82. Motion compensation unit 72 may also be configured to inter-predict depth blocks of a depth map.

[0093] As will be appreciated by those in the art, the coding system 10 of FIG. 1A is suitable for implementing various video coding or compression techniques. Some video compression techniques, such as inter prediction, intra prediction,  
25 and loop filters, have demonstrated to be effective. Therefore, the video compression techniques have been adopted into various video coding standards, such as H.264/AVC and H.265/HEVC.

[0094] Various coding tools such as adaptive motion vector prediction (AMVP) and merge mode (MERGE) are used to predict motion vectors (MVs) and  
30 enhance inter prediction efficiency and, therefore, the overall video compression efficiency.

[0095] The MVs noted above are utilized in bi-prediction. In a bi-prediction operation, two prediction blocks are formed. One prediction block is formed

5 using a MV of list0 (referred to herein as MV0). Another prediction block is formed using a MV of list1 (referred to herein as MV1). The two prediction blocks are then combined (e.g., averaged) in order to form a single prediction signal (e.g., a prediction block or a predictor block).

[0096] Other variations of the video decoder 30 can be used to decode the  
10 compressed bitstream. For example, the decoder 30 can produce the output video stream without the loop filtering unit. For example, a non-transform based decoder 30 can inverse-quantize the residual signal directly without the inverse-transform processing unit 78 for certain blocks or frames. In another implementation, the video decoder 30 can have the inverse-quantization unit 76  
15 and the inverse-transform processing unit 78 combined into a single unit.

[0097] FIG. 4 is a schematic diagram of a network device 400 (e.g., a coding device) according to an embodiment of the disclosure. The network device 400 is suitable for implementing the disclosed embodiments as described herein. In an embodiment, the network device 400 may be a decoder such as video decoder  
20 30 of FIG. 1A or an encoder such as video encoder 20 of FIG. 1A. In an embodiment, the network device 400 may be one or more components of the video decoder 30 of FIG. 1A or the video encoder 20 of FIG. 1A as described above.

[0098] The network device 400 comprises ingress ports 410 and receiver units  
25 (Rx) 420 for receiving data; a processor, logic unit, or central processing unit (CPU) 430 to process the data; transmitter units (Tx) 440 and egress ports 450 for transmitting the data; and a memory 460 for storing the data. The network device 400 may also comprise optical-to-electrical (OE) components and electrical-to-optical (EO) components coupled to the ingress ports 410, the  
30 receiver units 420, the transmitter units 440, and the egress ports 450 for egress or ingress of optical or electrical signals.

[0099] The processor 430 is implemented by hardware and software. The processor 430 may be implemented as one or more CPU chips, cores (e.g., as a

5 multi-core processor), FPGAs, ASICs, and DSPs. The processor 430 is in communication with the ingress ports 410, receiver units 420, transmitter units 440, egress ports 450, and memory 460. The processor 430 comprises a coding module 470. The coding module 470 implements the disclosed embodiments described above. For instance, the coding module 470 implements, processes,  
10 prepares, or provides the various coding operations. The inclusion of the coding module 470 therefore provides a substantial improvement to the functionality of the network device 400 and effects a transformation of the network device 400 to a different state. Alternatively, the coding module 470 is implemented as instructions stored in the memory 460 and executed by the processor 430.

15 **[00100]** The memory 460 comprises one or more disks, tape drives, and solid-state drives and may be used as an over-flow data storage device, to store programs when such programs are selected for execution, and to store instructions and data that are read during program execution. The memory 460 may be volatile and/or non-volatile and may be read-only memory (ROM),  
20 random access memory (RAM), ternary content-addressable memory (TCAM), and/or static random-access memory (SRAM).

**[00101]** Fig. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from Fig. 1A according to an exemplary embodiment. The apparatus 500 can  
25 implement techniques of this present application. The apparatus 500 can be in the form of a computing system including multiple computing devices, or in the form of a single computing device, for example, a mobile phone, a tablet computer, a laptop computer, a notebook computer, a desktop computer, and the like.

30 **[00102]** A processor 502 in the apparatus 500 can be a central processing unit. Alternatively, the processor 502 can be any other type of device, or multiple devices, capable of manipulating or processing information now-existing or hereafter developed. Although the disclosed implementations can be practiced

5 with a single processor as shown, e.g., the processor 502, advantages in speed and efficiency can be achieved using more than one processor.

[00103] A memory 504 in the apparatus 500 can be a read only memory (ROM) device or a random access memory (RAM) device in an implementation. Any other suitable type of storage device can be used as the memory 504. The  
10 memory 504 can include code and data 506 that is accessed by the processor 502 using a bus 512. The memory 504 can further include an operating system 508 and application programs 510, the application programs 510 including at least one program that permits the processor 502 to perform the methods described here. For example, the application programs 510 can include applications 1  
15 through N, which further include a video coding application that performs the methods described here. The apparatus 500 can also include additional memory in the form of a secondary storage 514, which can, for example, be a memory card used with a mobile computing device. Because the video communication sessions may contain a significant amount of information, they can be stored in  
20 whole or in part in the secondary storage 514 and loaded into the memory 504 as needed for processing.

[00104] The apparatus 500 can also include one or more output devices, such as a display 518. The display 518 may be, in one example, a touch sensitive display that combines a display with a touch sensitive element that is operable to  
25 sense touch inputs. The display 518 can be coupled to the processor 502 via the bus 512. Other output devices that permit a user to program or otherwise use the apparatus 500 can be provided in addition to or as an alternative to the display 518. When the output device is or includes a display, the display can be implemented in various ways, including by a liquid crystal display (LCD), a  
30 cathode-ray tube (CRT) display, a plasma display or light emitting diode (LED) display, such as an organic LED (OLED) display.

[00105] The apparatus 500 can also include or be in communication with an image-sensing device 520, for example a camera, or any other image-sensing

5 device 520 now existing or hereafter developed that can sense an image such as the image of a user operating the apparatus 500. The image-sensing device 520 can be positioned such that it is directed toward the user operating the apparatus 500. In an example, the position and optical axis of the image-sensing device 520 can be configured such that the field of vision includes an area that is  
10 directly adjacent to the display 518 and from which the display 518 is visible.

[00106] The apparatus 500 can also include or be in communication with a sound-sensing device 522, for example a microphone, or any other sound-sensing device now existing or hereafter developed that can sense sounds near the apparatus 500. The sound-sensing device 522 can be positioned such that it  
15 is directed toward the user operating the apparatus 500 and can be configured to receive sounds, for example, speech or other utterances, made by the user while the user operates the apparatus 500.

[00107] Although FIG. 5 depicts the processor 502 and the memory 504 of the apparatus 500 as being integrated into a single unit, other configurations can be  
20 utilized. The operations of the processor 502 can be distributed across multiple machines (each machine having one or more of processors) that can be coupled directly or across a local area or other network. The memory 504 can be distributed across multiple machines such as a network-based memory or memory in multiple machines performing the operations of the apparatus 500.  
25 Although depicted here as a single bus, the bus 512 of the apparatus 500 can be composed of multiple buses. Further, the secondary storage 514 can be directly coupled to the other components of the apparatus 500 or can be accessed via a network and can comprise a single integrated unit such as a memory card or multiple units such as multiple memory cards. The apparatus 500 can thus be  
30 implemented in a wide variety of configurations.

[00108] The present disclosure related to inter-picture prediction, while inter-picture prediction makes use of the temporal correlation between pictures in

5        order to derive a motion-compensated prediction (MCP) for a block of image samples.

[00109] For block-based MCP, a video picture is divided into rectangular blocks. Assuming homogeneous motion inside one block and that moving objects are larger than one block, for each block, a corresponding block in a  
10        previously decoded picture can be found that serves as a predictor. The general concept of MCP based on a translational motion model is illustrated in Fig.6. Using a translational motion model, the position of the block in a previously decoded picture is indicated by a motion vector  $(\Delta x, \Delta y)$ , where  $\Delta x$  specifies the horizontal and  $\Delta y$  specifies the vertical displacement relative to the position of  
15        the current block. The motion vector  $(\Delta x, \Delta y)$  could be of fractional sample accuracy to more accurately capture the movement of the underlying object. Interpolation is applied on the reference pictures to derive the prediction signal when the corresponding motion vector has fractional sample accuracy. The previously decoded picture is referred to as the reference picture and indicated  
20        by a reference index  $\Delta t$  to a reference picture list. These translational motion model parameters, i.e. motion vectors and reference indices, are further referred to as motion data. Two kinds of inter-picture prediction are allowed in modern video coding standards, namely uni-prediction and bi-prediction.

[00110] In case of bi-prediction, two sets of motion data, i.e.,  $(\Delta x_0, \Delta y_0, \Delta t_0)$  and  $(\Delta x_1, \Delta y_1, \Delta t_1)$ , are used to generate two MCPs (possibly from different  
25        pictures), which are then combined to get the final MCP. Per default, this is done by averaging but in case of weighted prediction, different weights can be applied to different MCPs, e.g., to compensate for scene fade outs. The reference pictures that can be used in bi-prediction are stored in two separate lists, namely  
30        list 0 and list 1. In order to limit the memory bandwidth in slices allowing bi-prediction, the HEVC standard restricts PUs with  $4 \times 8$  and  $8 \times 4$  luma prediction blocks to use uni-prediction only. Motion data is derived at the encoder using a motion estimation process. Motion estimation is not specified within video

5 standards so different encoders can utilize different complexity-quality tradeoffs in their implementations.

[00111] An overview block diagram of the HEVC inter-picture prediction is shown in Fig.7. The motion data of a block is correlated with the neighboring blocks. To exploit this correlation, motion data is not directly coded in the  
10 bitstream but predictively coded based on neighboring motion data. In HEVC, two concepts are used for that. The predictive coding of the motion vectors was improved in HEVC by introducing a new tool called advanced motion vector prediction (AMVP) where the best predictor for each motion block is signaled to the decoder. In addition, a new technique called inter-prediction block merging  
15 derives all motion data of a block from the neighboring blocks replacing the direct and skip modes in H.264/AVC.

[00112] Different kinds of inter prediction methods are implemented in the motion data coding module. Generally, the methods are called as inter prediction modes. Several inter prediction modes are introduced as following.

20 [00113] One of the inter prediction mode is AMVP (Advanced Motion Vector Prediction). As in previous video coding standards, the HEVC motion vectors are coded in terms of horizontal (x) and vertical (y) components as a difference to a so called motion vector predictor (MVP). The calculation of both motion vector difference (MVD) components is shown in Eq. (1.1) and (1.2).

$$25 \quad \text{MVD}_X = \Delta x - \text{MVP}_X \quad (1.1)$$

$$\text{MVD}_Y = \Delta y - \text{MVP}_Y \quad (1.2)$$

[00114] Motion vectors of the current block are usually correlated with the motion vectors of neighboring blocks in the current picture or in the earlier coded pictures. This is because neighboring blocks are likely to correspond to  
30 the same moving object with similar motion and the motion of the object is not likely to change abruptly over time. Consequently, using the motion vectors in neighboring blocks as predictors reduces the size of the signaled motion vector difference. The MVPs are usually derived from already decoded motion vectors

5 from spatial neighboring blocks or from temporally neighboring blocks in the co-located picture. In some cases, the zero motion vector can also be used as MVP. In H.264/AVC, this is done by doing a component wise median of three spatially neighboring motion vectors. Using this approach, no signaling of the predictor is required. Temporal MVPs from a co-located picture are only  
10 considered in the so called temporal direct mode of H.264/AVC. The H.264/AVC direct modes are also used to derive other motion data than the motion vectors.

[00115] In HEVC, the approach of implicitly deriving the MVP was replaced by a technique known as motion vector competition, which explicitly signals  
15 which MVP from a list of MVPs, is used for motion vector derivation. The variable coding quadtree block structure in HEVC can result in one block having several neighboring blocks with motion vectors as potential MVP candidates. The initial design of AMVP includes five MVPs from three different classes of predictors: three motion vectors from spatial neighbors, the median of the three  
20 spatial predictors and a scaled motion vector from a co-located, temporally neighboring block. Furthermore, the list of predictors was modified by reordering to place the most probable motion predictor in the first position and by removing redundant candidates to assure minimal signaling overhead. Then, significant simplifications of the AMVP design are developed such as removing  
25 the median predictor, reducing the number of candidates in the list from five to two, fixing the candidate order in the list and reducing the number of redundancy checks. The final design of the AMVP candidate list construction includes the following two MVP candidates: a. up to two spatial candidate MVPs that are derived from five spatial neighboring blocks; b. one temporal  
30 candidate MVPs derived from two temporal, co-located blocks when both spatial candidate MVPs are not available or they are identical; c. zero motion vectors when the spatial, the temporal or both candidates are not available.

5 [00116] As already mentioned, two spatial MVP candidates A and B are derived from five spatially neighboring blocks which are shown in the right part of Fig.8. The locations of the spatial candidate blocks are the same for both AMVP and inter-prediction block merging. The derivation process flow for the two spatial candidates A and B is depicted in Fig.9. For candidate A, motion data from the two blocks A0 and A1 at the bottom left corner is taken into account in a two pass approach. In the first pass, it is checked whether any of the candidate blocks contain a reference index that is equal to the reference index of the current block. The first motion vector found will be taken as candidate A. When all reference indices from A0 and A1 are pointing to a different reference picture than the reference index of the current block, the associated motion vector cannot be used as is. Therefore, in a second pass, the motion vectors need to be scaled according to the temporal distances between the candidate reference picture and the current reference picture. Eq. (1.3) shows how the candidate motion vector  $mv_{cand}$  is scaled according to a scale factor. ScaleFactor is calculated based on the temporal distance between the current picture and the reference picture of the candidate block  $td$  and the temporal distance between the current picture and the reference picture of the current block  $tb$ . The temporal distance is expressed in terms of difference between the picture order count (POC) values which define the display order of the pictures. The scaling operation is basically the same scheme that is used for the temporal direct mode in H.264/AVC. This factoring allows pre-computation of ScaleFactor at slice-level since it only depends on the reference picture list structure signaled in the slice header. Note that the MV scaling is only performed when the current reference picture and the candidate reference picture are both short term reference pictures. Parameter  $td$  is defined as the POC difference between the co-located picture and the reference picture of the co-located candidate block.

$$mv = \text{sign}(mv_{cand} \cdot \text{ScaleFactor}) \cdot ((|mv_{cand} \cdot \text{ScaleFactor}| + 2^7) >> 8) \quad (1.3)$$

$$5 \quad \text{ScaleFactor} = \text{clip}(-2^{12}, 2^{12}-1, (\text{tb} \cdot \text{tx} + 2^5) \gg 6) \quad (1.4)$$

$$\text{tx} = \frac{2^{14} + \left\lfloor \frac{\text{td}}{s} \right\rfloor}{\text{td}} \quad (1.5)$$

10 [00117] For candidate B, the candidates B0 to B2 are checked sequentially in the same way as A0 and A1 which were checked in the first pass. The second pass, however, is only performed when blocks A0 and A1 do not contain any motion information, i.e. are not available or coded using intra-picture prediction. Then, candidate A is set equal to the non-scaled candidate B, if found, and candidate B is set equal to a second, non-scaled or scaled variant of candidate B.

15 The second pass searches for non-scaled as well as for scaled MVs derived from candidates B0 to B2. Overall, this design allows to process A0 and A1 independently from B0, B1, and B2. The derivation of B should only be aware of the availability of both A0 and A1 in order to search for a scaled or an additional non-scaled MV derived from B0 to B2. This dependency is acceptable

20 given that it significantly reduces the complex motion vector scaling operations for candidate B. Reducing the number of motion vector scaling represents a significant complexity reduction in the motion vector predictor derivation process.

25 [00118] In HEVC, the block to the bottom right and at the center of the current block have been determined to be the most suitable to provide a good temporal motion vector predictor(TMVP). These candidates are illustrated in the left part of Fig.8 where C0 represents the bottom right neighbor and C1 represents the center block. Here again, motion data of C0 is considered first and, if not available, motion data from the co-located candidate block at the center is used

30 to derive the temporal MVP candidate C. The motion data of C0 is also considered as not being available when the associated PU belongs to a CTU beyond the current CTU row. This minimizes the memory bandwidth requirements to store the co-located motion data. In contrast to the spatial MVP

5 candidates, where the motion vectors may refer to the same reference picture, motion vector scaling is mandatory for the TMVP. Hence, the same scaling operation as for the spatial MVPs is used.

[00119] While the temporal direct mode in H.264/AVC always refers to the first reference picture in the second reference picture list, list 1, and is only  
10 allowed in bi-predictive slices, HEVC offers the possibility to indicate for each picture which reference picture is considered as the co-located picture. This is done by signaling in the slice header the co-located reference picture list and reference picture index as well as requiring that these syntax elements in all slices in a picture should specify the same reference picture.

15 [00120] Since the temporal MVP candidate introduces additional dependencies, it might be desirable to disable its usage for error robustness reasons. In H.264/AVC there is the possibility to disable the temporal direct mode for bi-predictive slices in the slice header (`direct_spatial_mv_pred_flag`). HEVC syntax extends this signaling by allowing to disable the TMVP at  
20 sequence level or at picture level (`sps/slice_temporal_mvp_enabled_flag`). Although the flag is signaled in the slice header, it is a requirement of bitstream conformance that its value shall be the same for all slices in one picture. Since the signaling of the picture-level flag depends on the SPS flag, signaling it in the PPS would introduce a parsing dependency between SPS and PPS. Another  
25 advantage of this slice header signaling is that if you want to change only the value of this flag and no other parameter in the PPS, there is no need to transmit a second PPS.

[00121] In general, motion data signaling in HEVC is similar as in H.264/AVC. An inter-picture prediction syntax element, `inter_pred_idc`, signals  
30 whether reference list 0, 1 or both are used. For each MCP obtained from one reference picture list, the corresponding reference picture ( $\Delta t$ ) is signaled by an index to the reference picture list, `ref_idx_l0/1`, and the MV ( $\Delta x$ ,  $\Delta y$ ) is represented by an index to the MVP, `mvp_l0/1_flag`, and its MVD. A newly

- 5 introduced flag in the slice header, `mvd_l1_zero_flag`, indicates whether the MVD for the second reference picture list is equal to zero and therefore not signaled in the bitstream. When the motion vector is fully reconstructed, a final clipping operation assures that the values of each component of the final motion vector will always be in the range of  $-2^{15}$  to  $2^{15}-1$ , inclusive.
- 10 **[00122]** Another one of the inter prediction modes is Inter-picture Prediction Block Merging. The AMVP list only contains motion vectors for one reference list while a merge candidate contains all motion data including the information whether one or two reference picture lists are used as well as a reference index and a motion vector for each list. Overall, the merge candidate list is constructed
- 15 based on the following candidates: a. up to four spatial merge candidates that are derived from five spatial neighboring blocks; b. one temporal merge candidate derived from two temporal, co-located blocks; c. additional merge candidates including combined bi-predictive candidates and zero motion vector candidates.
- [00123]** The first candidates in the merge candidate list are the spatial
- 20 neighbors. Up to four candidates are inserted in the merge list by sequentially checking A1, B1, B0, A0 and B2, in that order, according to the right part of Fig.8.
- [00124]** Instead of just checking whether a neighboring block is available and contains motion information, some additional redundancy checks are performed
- 25 before taking all the motion data of the neighboring block as a merge candidate. These redundancy checks can be divided into two categories for two different purposes: a. avoid having candidates with redundant motion data in the list; b. prevent merging two partitions that could be expressed by other means which would create redundant syntax.
- 30 **[00125]** When N is the number of spatial merge candidates, a complete redundancy check would consist of  $\frac{N \cdot (N-1)}{2}$  motion data comparisons. In case of the five potential spatial merge candidates, ten motion data comparisons would be needed to assure that all candidates in the merge list have different motion

5 data. During the development of HEVC, the checks for redundant motion data have been reduced to a subset in a way that the coding efficiency is kept while the comparison logic is significantly reduced. In the final design, no more than two comparisons are performed per candidate resulting in five overall comparisons. Given the order of {A1, B1, B0, A0, B2}, B0 only checks B1, A0  
10 only A1 and B2 only A1 and B1. In an embodiment of the partitioning redundancy check, the bottom PU of a  $2N \times N$  partitioning is merged with the top one by choosing candidate B1. This would result in one CU with two PUs having the same motion data which could be equally signaled as a  $2N \times 2N$  CU. Overall, this check applies for all second PUs of the rectangular and asymmetric  
15 partitions  $2N \times N$ ,  $2N \times nU$ ,  $2N \times nD$ ,  $N \times 2N$ ,  $nR \times 2N$  and  $nL \times 2N$ . It is noted that for the spatial merge candidates, only the redundancy checks are performed and the motion data is copied from the candidate blocks as it is. Hence, no motion vector scaling is needed here.

[00126] The derivation of the motion vectors for the temporal merge candidate  
20 is the same as for the TMVP. Since a merge candidate comprises all motion data and the TMVP is only one motion vector, the derivation of the whole motion data only depends on the slice type. For bi-predictive slices, a TMVP is derived for each reference picture list. Depending on the availability of the TMVP for each list, the prediction type is set to bi-prediction or to the list for which the  
25 TMVP is available. All associated reference picture indices are set equal to zero. Consequently for uni-predictive slices, only the TMVP for list 0 is derived together with the reference picture index equal to zero.

[00127] When at least one TMVP is available and the temporal merge  
candidate is added to the list, no redundancy check is performed. This makes the  
30 merge list construction independent of the co-located picture which improves error resilience. Consider the case where the temporal merge candidate would be redundant and therefore not included in the merge candidate list. In the event of a lost co-located picture, the decoder could not derive the temporal candidates

5 and hence not check whether it would be redundant. The indexing of all subsequent candidates would be affected by this.

[00128] For parsing robustness reasons, the length of the merge candidate list is fixed. After the spatial and the temporal merge candidates have been added, it can happen that the list has not yet the fixed length. In order to compensate for the coding efficiency loss that comes along with the non-length adaptive list index signaling, additional candidates are generated. Depending on the slice type, up to two kind of candidates are used to fully populate the list: a. Combined bi-predictive candidates; b. Zero motion vector candidates.

[00129] In bi-predictive slices, additional candidates can be generated based on the existing ones by combining reference picture list 0 motion data of one candidate with and the list 1 motion data of another one. This is done by copying  $\Delta x_0$ ,  $\Delta y_0$ ,  $\Delta t_0$  from one candidate, e.g. the first one, and  $\Delta x_1$ ,  $\Delta y_1$ ,  $\Delta t_1$  from another, e.g. the second one. The different combinations are predefined and given in Table 1.1.

Table 1.1

| Combination Order                                     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|-------------------------------------------------------|---|---|---|---|---|---|---|---|---|---|----|----|
| $\Delta x_0$ , $\Delta y_0$ , $\Delta t_0$ from Cand. | 0 | 1 | 0 | 2 | 1 | 2 | 0 | 3 | 1 | 3 | 2  | 3  |
| $\Delta x_1$ , $\Delta y_1$ , $\Delta t_1$ from Cand. | 1 | 0 | 2 | 0 | 2 | 1 | 3 | 0 | 3 | 1 | 3  | 2  |

[00130] When the list is still not full after adding the combined bi-predictive candidates, or for uni-predictive slices, zero motion vector candidates are calculated to complete the list. All zero motion vector candidates have one zero displacement motion vector for uni-predictive slices and two for bi-predictive slices. The reference indices are set equal to zero and are incremented by one for each additional candidate until the maximum number of reference indices is reached. If that is the case and there are still additional candidates missing, a reference index equal to zero is used to create these. For all the additional

5 candidates, no redundancy checks are performed as it turned out that omitting these checks will not introduce a coding efficiency loss.

[00131] For each PU coded in inter-picture prediction mode, a so called merge\_flag indicates that block merging is used to derive the motion data. The merge\_idx further determines the candidate in the merge list that provides all the  
10 motion data needed for the MCP. Besides this PU-level signaling, the number of candidates in the merge list is signaled in the slice header. Since the default value is five, it is represented as a difference to five (five\_minus\_max\_num\_merge\_cand). That way, the five is signaled with a short codeword for the 0 whereas using only one candidate, is signaled with a longer  
15 codeword for the 4. Regarding the impact on the merge candidate list construction process, the overall process remains the same although it terminates after the list contains the maximum number of merge candidates. In the initial design, the maximum value for the merge index coding was given by the number of available spatial and temporal candidates in the list. When e.g. only two  
20 candidates are available, the index can be efficiently coded as a flag. But, in order to parse the merge index, the whole merge candidate list has to be constructed to know the actual number of candidates. Assuming unavailable neighboring blocks due to transmission errors, it would not be possible to parse the merge index anymore.

25 [00132] A crucial application of the block merging concept in HEVC is its combination with a skip mode. In previous video coding standards, the skip mode was used to indicate for a block that the motion data is inferred instead of explicitly signaled and that the prediction residual is zero, i.e. no transform coefficients are transmitted. In HEVC, at the beginning of each CU in an inter-  
30 picture prediction slice, a skip\_flag is signaled that implies the following: a. the CU only contains one PU (2N×2N partition type); b. the merge mode is used to derive the motion data (merge\_flag equal to 1); c. no residual data is present in the bitstream.

5       **[00133]** A parallel merge estimation level was introduced in HEVC that indicates the region in which merge candidate lists can be independently derived by checking whether a candidate block is located in that merge estimation region (MER). A candidate block that is in the same MER is not included in the merge candidate list. Hence, its motion data does not need to be available at the time of  
10       the list construction. When this level is e.g. 32, all prediction units in a  $32 \times 32$  area can construct the merge candidate list in parallel since all merge candidates that are in the same  $32 \times 32$  MER, are not inserted in the list. As shown in Fig.10, there is a CTU partitioning with seven CUs and ten PUs. All potential merge candidates for the first PU0 are available because they are outside the first  $32 \times 32$   
15       MER. For the second MER, merge candidate lists of PUs 2-6 cannot include motion data from these PUs when the merge estimation inside that MER should be independent. Therefore, when looking at a PU5 for example, no merge candidates are available and hence not inserted in the merge candidate list. In that case, the merge list of PU5 consists only of the temporal candidate (if  
20       available) and zero MV candidates. In order to enable an encoder to trade-off parallelism and coding efficiency, the parallel merge estimation level is adaptive and signaled as `log2_parallel_merge_level_minus2` in the picture parameter set.

25       **[00134]** Another one of the inter prediction modes is Sub-CU based motion vector prediction. During the development of the new video coding technique, with QTBT, each CU can have at most one set of motion parameters for each prediction direction. Two sub-CU level motion vector prediction methods are considered in the encoder by splitting a large CU into sub-CUs and deriving motion information for all the sub-CUs of the large CU. Alternative temporal motion vector prediction (ATMVP) method allows each CU to fetch multiple  
30       sets of motion information from multiple blocks smaller than the current CU in the collocated reference picture. In spatial-temporal motion vector prediction (STMVP) method motion vectors of the sub-CUs are derived recursively by

5 using the temporal motion vector predictor and spatial neighboring motion vector.

[00135] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[00136] While the Sub-CU based motion vector prediction includes  
10 Alternative temporal motion vector prediction, Spatial-temporal motion vector prediction, Combined with Merge mode, and Pattern matched motion vector derivation. Which will be introduced in the following:

[00137] In the alternative temporal motion vector prediction (ATMVP) method, the motion vectors temporal motion vector prediction (TMVP) is  
15 modified by fetching multiple sets of motion information (including motion vectors and reference indices) from blocks smaller than the current CU. As shown in Fig. 11, the sub-CUs are square  $N \times N$  blocks ( $N$  is set to 4 by default).

[00138] ATMVP predicts the motion vectors of the sub-CUs within a CU in two steps. The first step is to identify the corresponding block in a reference  
20 picture with a so-called temporal vector. The reference picture is called the motion source picture. The second step is to split the current CU into sub-CUs and obtain the motion vectors as well as the reference indices of each sub-CU from the block corresponding to each sub-CU, as shown in Figure 6.

[00139] In the first step, a reference picture and the corresponding block is  
25 determined by the motion information of the spatial neighboring blocks of the current CU. To avoid the repetitive scanning process of neighboring blocks, the first merge candidate in the merge candidate list of the current CU is used. The first available motion vector as well as its associated reference index are set to be the temporal vector and the index to the motion source picture. This way, in  
30 ATMVP, the corresponding block may be more accurately identified, compared with TMVP, wherein the corresponding block (sometimes called collocated block) is always in a bottom-right or center position relative to the current CU.

5       **[00140]** In the second step, a corresponding block of the sub-CU is identified by the temporal vector in the motion source picture, by adding the temporal vector to the coordinate of the current CU. For each sub-CU, the motion information of its corresponding block (the smallest motion grid that covers the center sample) is used to derive the motion information for the sub-CU. After  
10       the motion information of a corresponding  $N \times N$  block is identified, it is converted to the motion vectors and reference indices of the current sub-CU, in the same way as TMVP of HEVC, wherein motion scaling and other procedures apply. For example, the decoder checks whether the low-delay condition (i.e. the POCs of all reference pictures of the current picture are smaller than the POC of  
15       the current picture) is fulfilled and possibly uses motion vector  $MV_x$  (the motion vector corresponding to reference picture list X) to predict motion vector  $MV_y$  (with X being equal to 0 or 1 and Y being equal to  $1-X$ ) for each sub-CU.

20       **[00141]** In the Spatial-temporal motion vector prediction method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. As shown in Fig. 12, it is considered that an  $8 \times 8$  CU which contains four  $4 \times 4$  sub-CUs A, B, C, and D. The neighboring  $4 \times 4$  blocks in the current frame are labelled as a, b, c, and d.

25       **[00142]** The motion derivation for sub-CU A starts by identifying its two spatial neighbors. The first neighbor is the  $N \times N$  block above sub-CU A (block c). If this block c is not available or is intra coded the other  $N \times N$  blocks above sub-CU A are checked (from left to right, starting at block c). The second neighbor is a block to the left of the sub-CU A (block b). If block b is not available or is intra coded other blocks to the left of sub-CU A are checked (from top to bottom,  
30       starting at block b). The motion information obtained from the neighboring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A is derived by following the same procedure of TMVP derivation as specified in HEVC. The

- 5 motion information of the collocated block at location D is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors (up to 3) are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.
- 10 [00143] In the Combined with Merge mode, the sub-CU modes are enabled as additional merge candidates and there is no additional syntax element required to signal the modes. Two additional merge candidates are added to merge candidates list of each CU to represent the ATMVP mode and STMVP mode. Up to seven merge candidates are used, if the sequence parameter set indicates
- 15 that ATMVP and STMVP are enabled. The encoding logic of the additional merge candidates is the same as for the merge candidates in HM, which means, for each CU in P or B slice, two more RD checks is needed for the two additional merge candidates.
- [00144] Pattern matched motion vector derivation (PMMVD) mode is based
- 20 on Frame-Rate Up Conversion (FRUC) techniques. With this mode, motion information of a block is not signaled but derived at decoder side.
- [00145] A FRUC flag is signaled for a CU when its merge flag is true. When the FRUC flag is false, a merge index is signaled and the regular merge mode is used. When the FRUC flag is true, an additional FRUC mode flag is signaled to
- 25 indicate which method (bilateral matching or template matching) is to be used to derive motion information for the block.
- [00146] At encoder side, the decision on whether using FRUC merge mode for a CU is based on RD cost selection as done for normal merge candidate. That is the two matching modes (bilateral matching and template matching) are both
- 30 checked for a CU by using RD cost selection. The one leading to the minimal cost is further compared to other CU modes. If a FRUC matching mode is the most efficient one, FRUC flag is set to true for the CU and the related matching mode is used.

5 [00147] Motion derivation process in FRUC merge mode has two steps. A  
 CU-level motion search is first performed, then followed by a Sub-CU level  
 motion refinement. At CU level, an initial motion vector is derived for the whole  
 CU based on bilateral matching or template matching. First, a list of MV  
 candidates is generated and the candidate which leads to the minimum matching  
 10 cost is selected as the starting point for further CU level refinement. Then a local  
 search based on bilateral matching or template matching around the starting  
 point is performed and the MV results in the minimum matching cost is taken as  
 the MV for the whole CU. Subsequently, the motion information is further  
 refined at sub-CU level with the derived CU motion vectors as the starting points.  
 15 [00148] For example, the following derivation process is performed for a  
 $W \times H$  CU motion information derivation. At the first stage, MV for the whole  
 $W \times H$  CU is derived. At the second stage, the CU is further split into  $M \times M$   
 sub-CUs. The value of  $M$  is calculated as in Eq. (1.8),  $D$  is a predefined splitting  
 depth which is set to 3 by default in the JEM. Then the MV for each sub-CU is  
 20 derived.

$$M = \max\{4, \min\{\frac{W}{2^D}, \frac{H}{2^D}\}\} \quad (1.8)$$

[00149] As shown in the Fig. 13, the bilateral matching is used to derive  
 motion information of the current CU by finding the closest match between two  
 blocks along the motion trajectory of the current CU in two different reference  
 25 pictures. Under the assumption of continuous motion trajectory, the motion  
 vectors MV0 and MV1 pointing to the two reference blocks shall be  
 proportional to the temporal distances, i.e., TD0 and TD1, between the current  
 picture and the two reference pictures. When the current picture is temporally  
 between the two reference pictures and the temporal distance from the current  
 picture to the two reference pictures is the same, the bilateral matching becomes  
 30 mirror based bi-directional MV.

[00150] In the bilateral matching merge mode, bi-prediction is always applied  
 since the motion information of a CU is derived based on the closest match

5 between two blocks along the motion trajectory of the current CU in two different reference pictures. There is no such limitation for the template matching merge mode. In the template matching merge mode, the encoder can choose among uni-prediction from list0, uni-prediction from list1 or bi-prediction for a CU. The selection is based on a template matching cost as follows:

10                                   If  $\text{costBi} \leq \text{factor} * \min(\text{cost0}, \text{cost1})$   
                                           bi-prediction is used;  
                                   Otherwise, if  $\text{cost0} \leq \text{cost1}$   
                                           uni-prediction from list0 is used;  
 15                                   Otherwise,  
                                           uni-prediction from list1 is used;  
                                   where cost0 is the SAD of list0 template matching, cost1 is the SAD of list1 template matching and costBi is the SAD of bi-prediction template matching. The value of factor is equal to 1.25, which means that the selection process is biased toward bi-prediction. The inter prediction direction selection is only applied to the CU-level template matching process.

25                   [00151] As shown in Fig. 14, template matching is used to derive motion information of the current CU by finding the closest match between a template (top and/or left neighbouring blocks of the current CU) in the current picture and a block (same size to the template) in a reference picture. Except the aforementioned FRUC merge mode, the template matching is also applied to AMVP mode. With template matching method, a new candidate is derived. If the newly derived candidate by template matching is different to the first existing AMVP candidate, it is inserted at the very beginning of the AMVP candidate list and then the list size is set to two (meaning remove the second existing AMVP candidate). When applied to AMVP mode, only CU level search is applied.

30

5       [00152] The MV candidate set at CU level consists of: a. Original AMVP candidates if the current CU is in AMVP mode; b. all merge candidates; c. several MVs in the interpolated MV field; d. top and left neighbouring motion vectors.

10       [00153] It is noted that the interpolated MV field mentioned above is generated before coding a picture for the whole picture based on unilateral ME. Then the motion field may be used later as CU level or sub-CU level MV candidates. First, the motion field of each reference pictures in both reference lists is traversed at  $4 \times 4$  block level. For each  $4 \times 4$  block, if the motion associated to the block passing through a  $4 \times 4$  block in the current picture, as shown in  
15       Fig.15, and the block has not been assigned any interpolated motion, the motion of the reference block is scaled to the current picture according to the temporal distance TD0 and TD1 (the same way as that of MV scaling of TMVP in HEVC) and the scaled motion is assigned to the block in the current frame. If no scaled MV is assigned to a  $4 \times 4$  block, the block's motion is marked as unavailable in  
20       the interpolated motion field.

      [00154] When using bilateral matching, each valid MV of a merge candidate is used as an input to generate a MV pair with the assumption of bilateral matching. For example, one valid MV of a merge candidate is (MVa, refa) at reference list A. Then the reference picture refb of its paired bilateral MV is  
25       found in the other reference list B so that refa and refb are temporally at different sides of the current picture. If such a refb is not available in reference list B, refb is determined as a reference which is different from refa and its temporal distance to the current picture is the minimal one in list B. After refb is determined, MVb is derived by scaling MVa based on the temporal distance  
30       between the current picture and refa, refb.

      [00155] Four MVs from the interpolated MV field are also added to the CU level candidate list. More specifically, the interpolated MVs at the position (0, 0), (W/2, 0), (0, H/2) and (W/2, H/2) of the current CU are added.

- 5       [00156] When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set.
- [00157] At the CU level, up to 15 MVs for AMVP CUs and up to 13 MVs for merge CUs are added to the candidate list.
- [00158] The MV candidate set at sub-CU level consists of: a. an MV  
10       determined from a CU-level search; b. top, left, top-left and top-right neighbouring MVs; c. scaled versions of collocated MVs from reference pictures; d. up to 4 ATMVP candidates; e. up to 4 STMVP candidates.
- [00159] The scaled MVs from reference pictures are derived as follows. All the reference pictures in both lists are traversed. The MVs at a collocated  
15       position of the sub-CU in a reference picture are scaled to the reference of the starting CU-level MV.
- [00160] ATMVP and STMVP candidates are limited to the four first ones.
- [00161] At the sub-CU level, up to 17 MVs are added to the candidate list.
- [00162] Motion vector can be refined by different methods combining with the  
20       different inter prediction modes.
- [00163] MV refinement is a pattern based MV search with the criterion of bilateral matching cost or template matching cost. In the current development, two search patterns are supported – an unrestricted center-biased diamond search (UCBDS) and an adaptive cross search for MV refinement at the CU level and  
25       sub-CU level, respectively. For both CU and sub-CU level MV refinement, the MV is directly searched at quarter luma sample MV accuracy, and this is followed by one-eighth luma sample MV refinement. The search range of MV refinement for the CU and sub-CU step are set equal to 8 luma samples.
- [00164] In bi-prediction operation, for the prediction of one block region, two  
30       prediction blocks, formed using a MV of list0 and a MV of list1, respectively, are combined to form a single prediction signal. In the decoder-side motion vector refinement (DMVR) method, the two motion vectors of the bi-prediction are further refined by a bilateral template matching process. The bilateral

5        template matching applied in the decoder to perform a distortion-based search between a bilateral template and the reconstruction samples in the reference pictures in order to obtain a refined MV without transmission of additional motion information.

10        [00165] In DMVR, a bilateral template is generated as the weighted combination (i.e. average) of the two prediction blocks, from the initial MV0 of list0 and MV1 of list1, respectively, as shown in Fig. 16. The template matching operation consists of calculating cost measures between the generated template and the sample region (around the initial prediction block) in the reference picture. For each of the two reference pictures, the MV that yields the minimum  
15        template cost is considered as the updated MV of that list to replace the original one. In the current development, nine MV candidates are searched for each list. The nine MV candidates include the original MV and 8 surrounding MVs with one luma sample offset to the original MV in either the horizontal or vertical direction, or both. Finally, the two new MVs, i.e., MV0' and MV1' as shown in  
20        Fig.16, are used for generating the final bi-prediction results. A sum of absolute differences (SAD) is used as the cost measure.

[00166] DMVR is applied for the merge mode of bi-prediction with one MV from a reference picture in the past and another from a reference picture in the future, without the transmission of additional syntax elements.

25        [00167] The usage of the TMVP, in AMVP as well as in the merge mode, requires the storage of the motion data (including motion vectors, reference indices and coding modes) in co-located reference pictures. Considering the granularity of motion representation, the memory size needed for storing motion data could be significant. HEVC employs motion data storage reduction (MDSR)  
30        to reduce the size of the motion data buffer and the associated memory access bandwidth by sub-sampling motion data in the reference pictures. While H.264/AVC is storing these information on a 4×4 block basis, HEVC uses a 16×16 block where, in case of sub-sampling a 4×4 grid, the information of the

5 top-left 4×4 block is stored. Due to this sub-sampling, MDSR impacts on the quality of the temporal prediction.

[00168] Furthermore, there is a tight correlation between the position of the MV used in the co-located picture, and the position of the MV stored by MDSR. During the standardization process of HEVC, it turned out that storing the motion data of the top left block inside the 16×16 area together with the bottom right and center TMVP candidates provide the best tradeoff between coding efficiency and memory bandwidth reduction.

[00169] In HEVC, motion vector accuracy is one-quarter pel (one-quarter luma sample and one-eighth chroma sample for 4:2:0 video). In the current development, the accuracy for the internal motion vector storage and the merge candidate increases to 1/16 pel. The higher motion vector accuracy (1/16 pel) is used in motion compensation inter prediction for the CU coded with skip/merge mode. For the CU coded with normal AMVP mode, either the integer-pel or quarter-pel motion is used.

20 [00170] When a motion vector points to a fractional sample position, motion compensated interpolation is needed. For the luma interpolation filtering, an 8-tap separable DCT-based interpolation filter is used for 2/4 precision samples and a 7-tap separable DCT-based interpolation filter is used for 1/4 precisions samples, as shown in Table 1.2.

25

Table 1.2

| Position | Filter coefficients                |
|----------|------------------------------------|
| 1/4      | { -1, 4, -10, 58, 17, -5, 1 }      |
| 2/4      | { -1, 4, -11, 40, 40, -11, 4, -1 } |
| 3/4      | { 1, -5, 17, 58, -10, 4, -1 }      |

[00171] Similarly, a 4-tap separable DCT-based interpolation filter is used for the chroma interpolation filter, as shown in Table 1.3.

Table 1.3

| Position | Filter coefficients |
|----------|---------------------|
| 1/8      | { -2, 58, 10, -2 }  |
| 2/8      | { -4, 54, 16, -2 }  |
| 3/8      | { -6, 46, 28, -4 }  |
| 4/8      | { -4, 36, 36, -4 }  |
| 5/8      | { -4, 28, 46, -6 }  |
| 6/8      | { -2, 16, 54, -4 }  |
| 7/8      | { -2, 10, 58, -2 }  |

5     **[00172]**   For the vertical interpolation for 4:2:2 and the horizontal and vertical interpolation for 4:4:4 chroma channels, the odd positions in Table 1.3 are not used, resulting in 1/4<sup>th</sup> chroma interpolation.

[00173] For the bi-directional prediction, the bit-depth of the output of the interpolation filter is maintained to 14-bit accuracy, regardless of the source bit-depth, before the averaging of the two prediction signals. The actual averaging  
10 process is done implicitly with the bit-depth reduction process as:

$$\text{predSamples}[x, y] = (\text{predSamplesL0}[x, y] + \text{predSamplesL1}[x, y] + \text{offset}) \gg \text{shift} \quad (1.9)$$

$$\text{shift} = 15 - \text{BitDepth} \quad (1.10)$$

$$\text{offset} = 1 \ll (\text{shift} - 1) \quad (1.11)$$

[00174] To reduce complexity, bi-linear interpolation instead of regular 8-tap  
20 HEVC interpolation is used for both bilateral matching and template matching.

[00175] The calculation of matching cost is a bit different at different steps. When selecting the candidate from the candidate set at the CU level, the matching cost is the SAD of bilateral matching or template matching. After the starting MV is determined, the matching cost  $\mathcal{C}$  of bilateral matching at sub-CU level search is calculated as follows:

5

$$C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|) \quad (1.12)$$

where  $w$  is a weighting factor which is empirically set to 4,  $MV$  and  $MV^s$  indicate the current MV and the starting MV, respectively. SAD is still  
 10 used as the matching cost of template matching at sub-CU level search.

[00176] In FRUC mode, MV is derived by using luma samples only. The derived motion will be used for both luma and chroma for MC inter prediction. After MV is decided, final MC is performed using 8-taps interpolation filter for luma and 4-taps interpolation filter for chroma.

15 [00177] In the absence of any decoder side motion vector derivation, the motion vectors and reference indices of coding units that are coded with any inter-coding mode are reconstructed or inferred without any pixel level operations on any coding unit within that frame. The differential coding of a motion vector using an appropriately scaled version of an already reconstructed motion vector of a spatial  
 20 or temporally co-located or interpolated neighbor as well as the process of inheriting a reconstructed motion vector through a merge process are computationally simple and hence the dependent reconstruction or inheritance process does not pose any major decoder side design complexity issue.

[00178] The decoder side motion vector refinement or pattern matched motion  
 25 vector derivation schemes proposed till now allow the refined motion vector(s) of a spatially neighboring coding unit to be employed as motion vector predictor(s) in the differential coding of the motion vector(s) of a current coding unit. This results in significant coding gains through either the reduction in motion vector delta (MVD) coding bits with the spatially neighboring CU's refinement MVs  
 30 when the current CU is INTER/AFFINE-INTER and through improved starting point for current CU's refinement when it is a CU that employs DMVR/PMMVD. However, this significantly impacts the concurrency in processing of the coding units of the current frame as the motion compensation or refinement followed by

5 motion compensation for a given coding unit cannot start until the final motion  
vector of the spatially neighboring CU on which the current CU depends on. Even  
one refinement based dependency in the chain will stall the processing pipeline till  
that dependency is resolved. Given that motion based partitioning can have a wide  
range of granularities (from a complete coded tree block being one CU to as small  
10 as a 4x4 CU), the sequential dependency results in reduced parallelism that will  
affect the timing within which the tasks of fetching the required reference data and  
performing decoder-side MV refinement and/or motion compensated prediction  
need to be completed. This will result in a significant over-design (e.g. higher  
clock, wider execution units, wider buses, etc.) to handle the worst-case timing  
15 and significant under-utilization of the designed execution units in the average  
cases.

[00179] On the other hand, when the dependency issue is resolved by forcing all  
CUs in the current access unit to not use the refined motion vector of any coding  
unit as a predictor or starting point for refinement, the coding gains suffer  
20 significantly. This is because the RDO process decides DMVR/PMMVD to be  
superior to the other inter-coding modes. But in the absence of the decoder-side  
refinement, the MVD coding bits increase significantly (when compared to the no  
DMVR/PMMVD case) and also the starting points for the refinements end up  
being inferior. Hence there is a significant compression loss by not using any  
25 refined MVs.

[00180] Hence there is a need for a method that arrives at a suitable trade-off  
between the coding loss and the complexity increase by allowing refined motion  
vectors within the current access unit to be used as a predictor or as a starting  
point for other refinements.

30 [00181] The proposed method and embodiments disclosed herein determines  
the availability of refined motion vectors of spatially neighboring coding units in  
such a way that a set of coding units or sub-coding units within a coded tree  
block can configure their data pre-fetch in a concurrent manner in a given stage

5 of a regular pipeline and also perform their refinement process in a concurrent manner in the next stage of that regular pipeline. Specifically, the concept of a lag between the top CTB row and current CTB row is utilized in determining such availability. Also, the concept of a concurrency set is introduced to normatively partition some coding units, if necessary, into sub-coding-units to  
10 meet the concurrency requirements of the pipeline. Compared to not using any refined motion vector from the current picture, the proposed approach provides a higher coding gain while ensuring that the dependency does not overly constrain the hardware implementation of the refinement process. Also, by pre-fetching around an unrefined motion vector and using a normative padding process to  
15 access samples that go outside the normative amount of pre-fetched samples, the pipeline latency is further reduced to make even left or top-right CTB refined MVs to be used for refinement of current CTB CUs. The process is also extended to finer granularities than CTB level.

[00182] Given that decoder side motion vector refinement/derivation is a  
20 normative aspect of a coding system, the encoder will also have to perform the same error surface technique in order to not have any drift between the encoder's reconstruction and the decoder's reconstruction. Hence, all aspects of all embodiments are applicable to both encoding and decoding systems.

[00183] All embodiments of the present invention are applicable to both  
25 PMMVD and DMVR methods.

#### **Embodiment 1**

[00184] By considering a regular CTB level pipeline design, each processing stage should be preceded by a data pre-fetch stage. Both the data fetch stage and processing stage should be able to use the entire time of the pipeline slot.  
30 Following Fig.17 shows such a regular pipeline at CTB level with DMA on the left and DMVR + MC on the right, corresponding to the data pre-fetch and processing stages.

5       [00185] In case of both top and current row starting at same time, doesn't provide the spatial refined motion vector from the neighbors. The following figure shows the CTB pipeline cross rows when both row processing starts at same time.

10       [00186] From Fig. 18, it is very clear that none of the CTB in the current row at the DMA stage have the top row DMVR+MC stage completed. This is similar to not using any refined motion vector from the current access unit for AMVP or as a starting motion vector for decoder-side motion refinement.

15       [00187] By introducing a lag of N CTBs ( $N > 0$ ) between the current CTB row and its Top neighbor CTB row, Current CTB gets the refined motion vectors of some of the top CTB neighbors. Following Fig. 19 shows the processing pipeline with  $N=2$ . In HEVC and AVC, intra prediction depends on completion of top right neighbor CTB. A lag value of  $N=2$  will now bring a similar dependency for inter prediction.

20       [00188] From Fig. 19, during the DMA or data pre-fetch stage of  $CN+1$ , both  $TN$  and  $TN+1$  DMVR+MC stage are completed, hence  $CN+1$  CTB can use the refined MVs of the Top and Top left CTBs for Inter MVP and as starting MVs for decoder-side MV refinement.

25       [00189] It should be noted that even with the concept of lag between consecutive CTB rows, any coding unit that is a spatial neighbor for a current CU within the current CTB is still considered unavailable. Also, given that data pre-fetch stage has been introduced as a pipeline stage, left and bottom-left neighbor CTBs are always considered as unavailable. Hence, this scheme is more beneficial when the maximum CTB size is smaller. This is summarized in the Table 3.1.

30

**Table 3.1 Final refined MV Availability status of spatial neighbors CTBs with lags 0, 2, and 3**

| Spatial | Availability |
|---------|--------------|
|---------|--------------|

| Neighbour CTB | LAG 0 | LAG 2 | LAG3 |
|---------------|-------|-------|------|
| Left          |       |       |      |
| Bottom-Left   |       |       |      |
| Top           |       |       |      |
| Top Left      |       |       |      |
| Top Right     |       |       |      |

5

**[00190]** By using the method disclosed in embodiment 1, improved coding gains due to increasing the availability of spatial refined motion vectors by introducing the CTB lag of N. “N” can be selected as per the design constraints of the video coding system.

10

#### **Embodiment 2**

15

**[00191]** In embodiment 1, all coding units at a CTB level were considered as part of a concurrency set such that their data-prefetches can happen concurrently during a pipeline stage while their refinement and motion compensation processing can happen concurrently during the following pipeline stage. In this embodiment, a concept of concurrency set that can exist at a sub-CTB level is introduced in order further improve the coding gains by being able to use the final refined motion vectors in more cases.

**[00192]** For instance, there are cases where a larger CTB is force partitioned into quad-tree partitioned coding units that are coded in a recursive z-scan order.

20

**[00193]** As in PMMVD, it is also possible to force partition a given coding unit into sub-coding units. A concurrency set is defined to be a set of pixels in a CTB that correspond to one partition of recursive quad-tree partition of the CTB. For instance, concurrency set can be chosen as 64x64 or 32x32 for a CTB size of 128x128. A given coding unit that spans across more than one concurrency set is force partitioned for decoder-side motion vector refinement purposes into as many sub coding units (sub-CUs) as the number of concurrency sets that it spans.

25

5 The dependency across these concurrency sets is assumed to be in a recursive z-scan order. Thus independent of the actual partitioning of the CTB into coding units, a concurrency set becomes an independent set of pixels, the processing for which can be performed concurrently to have a regular sub-CTB level pipeline that can have a data pre-fetch stage followed by a refinement and motion  
10 compensation stage in a manner similar to the CTB level pipeline in embodiment 1.

[00194] Fig.20a-c show the pipeline of using a 1-level deep quad-tree split of a CTB and with lag of 2 CTBs between two consecutive CTB rows.

[00195] From the above figures, the improvements in the availability of final  
15 refined spatial neighbor motion vectors can be seen. Concurrency set Z0 has all the neighbor concurrency sets available except bottom left, concurrency set Z1 has all top concurrency set neighbors available but left and bottom left concurrency set neighbors are not available, concurrency set Z2 has all concurrency set neighbors except Top right and bottom left concurrency sets,  
20 and concurrency set Z3 has only Top and top left concurrency sets available. This is summarized in the Table 3.2.

**Table 3.2 Refined spatial neighbor concurrency set availability status  
for 1-level deep quad-trees split with a lag of 2 CTBs between consecutive CTB  
rows**

25

| Spatial<br>Neighbours | Availability for lag2 in case of Sub<br>CTB level |    |    |    |
|-----------------------|---------------------------------------------------|----|----|----|
|                       | Z0                                                | Z1 | Z2 | Z3 |
| Left                  |                                                   |    |    |    |
| BottomLeft            |                                                   |    |    |    |
| Top                   |                                                   |    |    |    |
| Top Left              |                                                   |    |    |    |

|           |  |  |  |  |
|-----------|--|--|--|--|
| Top Right |  |  |  |  |
|-----------|--|--|--|--|

5

[00196] By using the method of embodiment 2, the forced partitioning of a CTB into concurrency sets for performing the decoder-side motion vector refinement in a manner that is independent of the actual partitioning of the CTB makes more final refined motion vectors to be available in conjunction with the concept of a configurable CTB lag between consecutive CTB rows. This helps improve the coding gains relative to embodiment 1 while still allowing for a regular pipeline that allows for a data pre-fetch stage that precedes the decoder-side motion vector refinement and motion compensated prediction stage.

[00197] Thus both embodiments 1 and 2 in the present disclosure may be summarized as, a method for determining the availability of the refined motion vectors of a spatially neighboring set, comprising at least one coding unit or sub-coding unit, for a current set, comprising at least one coding unit or sub coding unit, may comprise the steps of:

determining that the spatially neighboring set's refined motion vectors to be available when those refined motion vectors have been computed in a pipeline stage ahead of the data pre-fetch stage of the current set; and

partitioning a coding unit normatively into as many sub-coding units as the number of concurrency sets that the coding unit spans so that the data pre-fetch and decoder-side motion vector refinement and motion compensation for each sub-coding unit can happen independent of the other sub coding-units, but in a concurrent manner with other coding or sub-coding units that are part of a current concurrency set.

[00198] Fig. 22 describes different pattern directions, where a-f describes different patterns (in black) for different pixel situations (in black), and the details are shown in the Fig. 22.

[00199] Fig. 23 describes error surface, and the details are shown in the fig. 23. Terms T B L R and C in Figure 23 refer to relative directions Top Bottom Left

- 5 Right and Centre respectively. Moreover, an example of error surface cost combination is show in table 4.

Table 4

| Case No | Cost function used for integer distance refinement                                                                | Direction of movement                                                                                                                                            | Cost Function used for sub-pixel error surface technique                                                                                                                                                 |
|---------|-------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | DMVR Template cost<br>$T_{avg} = C_0 + C_1 / 2$<br>$Cost\ L_0 =  T_{avg} - L_0 $<br>$Cost\ L_1 =  T_{avg} - L_1 $ | 1. Move Jointly on L0/L1 with reflection of delta motion on L1/L0 with $T_{avg}$ updated with new centre of L0 and L1<br>2. Move on both L0 and L1 Independently | DMVR Template cost and Symmetrical Bilateral cost if centre is the best, else Integer pixel distance movement is allowed when cost used for error surface is different from integer distance refinement. |
| 2       | Symmetrical Bilateral cost<br>$Cost =  C_0' - C_1' $                                                              | Move Jointly on L0/L1 with reflection of delta motion on L1/L0                                                                                                   | DMVR Template cost and Symmetrical Bilateral cost if centre is the best, else Integer pixel distance movement is allowed when cost used for error surface is different from integer distance refinement. |
| 3       | Bilateral Cost<br>$Cost\ L_0 =  C_1 - C_0 $<br>$Cost\ L_1 =  C_0 - C_1 $                                          | Refine L0/L1 w.r.t L1/L0 centre and scale the best MVs based on the POC distance to move to                                                                      | DMVR Template cost and Symmetrical Bilateral cost if centre is the best, else Integer pixel distance movement is allowed when cost used for                                                              |

|   |                                                                                                               |                                                                          |                                                                 |
|---|---------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|-----------------------------------------------------------------|
|   |                                                                                                               | the new L1/L0<br>centre to get<br>template                               | error surface is different from<br>integer distance refinement. |
| 4 | Template Cost<br>$\text{Cost L0} =  \text{Tnbr} - \text{C0} $<br>$\text{Cost L1} =  \text{Tnbr} - \text{C1} $ | Uni directional<br>Refinement w.r.t<br>the neighbor pixel<br>as template | Template Cost                                                   |

5

**Embodiment 3**

[00200] The search range for decoder-side motion vector refinement increases the worst-case external memory accesses and also the internal memory buffers. To counter this, some prior art methods do not bring any additional samples that are based on the search range, but only use the samples required for performing motion compensation using the merge mode motion vectors. Additional samples required for refinement are obtained purely through motion compensated interpolation that employs padding for the unavailable sample with the last available sample before it. It is also possible to arrive at a trade-off between external memory bandwidth and padding introduced coding efficiency reduction by fetching one or more lines of samples beyond just the samples required for motion compensation using the merge mode motion vectors without refinement, but still less than what are required for covering the entire refinement range.

[00201] In this embodiment, with such padded motion compensation, the pre-fetch for a given CTB (or a sub-partition of a CTB at the first level) is performed using the unrefined motion vectors of coding units in the causal neighbor CTBs if the refined neighbor CTB motion vectors are not available at the time of pre-

5        fetch. The refined motion vector of a coding unit in the neighbor CTB can be used as the starting point for performing the refinement for a coding unit within the current CTB that merges with that coding unit in the neighbor CTB. Any unavailable samples relative to the pre-fetched data can be accessed or interpolated through padding. Thus, the use of padded samples obviates the need  
10        for pre-fetch to be performed only after the coding unit in the neighbor CTB completes its refinement, thus reducing the latency of the dependency. This method works reasonably well when the refinement search range iterations are low in number or when the refinement process exit early. The reduction of the pipeline latency implies that refined motion vectors of CTBs that are just one  
15        pipeline slot ahead of the current CTB can be used as the starting points for refinement in coding units within the current CTB. For example, even with a lag of 1 CTB between CTB rows, the top CTB's refined MVs can be used for refinement of coding units within the current CTB. Also, the left CTB's refined MVs can be used for refinement of coding units within the current CTB. Thus,  
20        barring the refined MVs of coding units within the current CTB, all other neighbor refined MVs can be employed to bring back the coding gains. The availabilities are summarized in Table 5 below.

25

30

5

**Table 5: Use of neighbor's refined and unrefined MVs for pre-fetch and refinement based on neighbor CTB's pipeline lag when CTB-row lag=2**

| With CTB-row lag of 2, if neighbor CU's CTB has a lag of | Pre-fetch based on                | Refinement start based on      |
|----------------------------------------------------------|-----------------------------------|--------------------------------|
| $\geq 2$ (i.e. top-left and top CTBs)                    | Neighbor's refined MV             | Neighbor's refined MV          |
| 1 (left and top-right CTBs)                              | Neighbor's start of refinement MV | Neighbor's refined MV          |
| 0 (within current CTB)                                   | Neighbor's pre-fetch MV           | Neighbor's refinement start MV |

10      **[00202]** Similar to embodiment 2, when each CTB is quad-tree split at the first depth of splitting, it is possible to have a pipeline of pre-fetch followed by refinement that is at the granularity of a quarter of a CTB (QCTB). In this case, with the pre-fetch based on unrefined neighbor QCTB MVs and with neighbor QCTB refined MVs as starting points for refinement with use of padding to get

15      unavailable samples, the z-scan order of encoding the 4 QCTB coding units, more refined MVs can be tapped for refinement while still ensuring that the pre-fetch followed by refinement pipeline at the QCTB level can work. Table 6 below summarizes the MVs of neighbor used by each of the QCTBs within a CTB.

20

25

5

**Table 6: Use of neighbor's refined and unrefined MVs for pre-fetch and refinement based on pipeline lag value of neighbor CU's QCTB.**

| if neighbor CU's QCTB<br>has a lag of | Pre-fetch based on                   | Refinement start based on         |
|---------------------------------------|--------------------------------------|-----------------------------------|
| $\geq 2$                              | Neighbor's refined MV                | Neighbor's refined MV             |
| 1                                     | Neighbor's start of refinement<br>MV | Neighbor's refined MV             |
| 0 (within current<br>QCTB)            | Neighbor's pre-fetch MV              | Neighbor's refinement<br>start MV |

10 [00203] This embodiment enables even the left, top-right, and bottom-left neighbor's refined MV to be used if the neighbor CU is in a neighbor CTB (or QCTB) and not within the current CTB (or QCTB). This improves the coding gains while still ensuring a regular pipeline where pre-fetch is performed at CTB (or QCTB) level and refinement of all coding units within the current CTB (or

15 QCTB) can be performed in parallel.

[00204] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed

20 by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable

25 media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal

5 or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

10

[00205] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

25  
30 [00206] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly,

5 the term “processor,” as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the

10 techniques could be fully implemented in one or more circuits or logic elements. [00207] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices

15 configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interpretable hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

20 [00208] While several embodiments have been provided in the present disclosure, it should be understood that the disclosed systems and methods might be embodied in many other specific forms without departing from the spirit or scope of the present disclosure. The present examples are to be considered as illustrative and not restrictive, and the intention is not to be limited to the details

25 given herein. For example, the various elements or components may be combined or integrated in another system or certain features may be omitted, or not implemented.

[00209] In addition, techniques, systems, subsystems, and methods described and illustrated in the various embodiments as discrete or separate may be

30 combined or integrated with other systems, modules, techniques, or methods without departing from the scope of the present disclosure. Other items shown or discussed as coupled or directly coupled or communicating with each other may be indirectly coupled or communicating through some interface, device, or intermediate component whether electrically, mechanically, or otherwise. Other

35 examples of changes, substitutions, and alterations are ascertainable by one skilled in the art and could be made without departing from the spirit and scope disclosed herein.

5

**CLAIMS**

1. A method of decoding for video data, such as in DMRV or PMMVD, implemented by a decoding unit or an encoding unit, comprising:

- determining, for a current set which comprises at least one coding unit (CU), or sub coding unit, that a refined motion vector of a spatially neighboring set is available when the refined motion vectors have been obtained, wherein the spatially neighboring set comprises at least one coding unit (CU) or sub-coding unit; and
- providing the said refined motion vector to a current processing stage of a processing pipeline for the current set.

15

2. The method according to claim 1, wherein

- a coding unit is partitioned into as many sub-coding units as the number of concurrency set(s) that the coding unit (CU) spans, so that the data pre-fetch and decoder-side motion vector refinement and motion compensation for each sub-coding unit happens independently of the other sub coding-units, but in a concurrent manner with other coding or sub-coding units that are part of a current concurrency set.

25

3. The method according to claim 1 or 2, wherein the method is performed in a data pre-fetch stage of a current set;

wherein the refined motion vectors have been computed in a pipeline stage of the current set, wherein the pipeline stage is an ahead stage of the data pre-fetch stage.

30

4. The method according to any preceding claim, comprising

- is a refined motion vector of a spatially neighboring set is not available at the time of pre-fetch of the current set, using unrefined motion vectors of coding units in casual neighbor CTBs; and / or

- 5     - using the refined motion vector of a coding unit in the neighbour CTB that does not fall within the same concurrency set, such as may merge with that coding unit, as start of refinement motion vector center;  
and / or  
using padded samples whenever the refinement requires samples that are not pre-  
10    fetched based on a configurable additional sample fetch range around the pre-fetch center.

5.     The method according to any preceding claim, wherein the refined motion vector is obtained by computing the refined motion vector in an earlier processing stage that is prior to the current processing stage, wherein the earlier processing stage is prior to the current processing stage by a number of processing stages of the processing pipeline.

6.     The method according to any preceding claim, wherein the number of processing stages earlier or ahead is determined according to a configured lag.

7.     The method according to claim 6, wherein the configured lag is an integer,  $N$ ,  $N$  is greater than or equal to 0; preferably  $N=1$ ; even more preferably  $N=2$  or more.

25    8.    The method according to any of claims 1-7, wherein the current processing stage is a pre-fetch stage e.g. DMA of the current set or a motion vector refinement and motion compensation processing stage e.g. DMVR+MC of the current set; and the earlier processing stage is a motion vector refinement and motion compensation processing stage e.g. DMVR+MC of the spatially neighboring set which obtains the  
30    refined motion vector of the spatially neighboring set.

- 5     9. The method according to any preceding claim, wherein a concurrency set comprises a set of independent pixels for which a processing stage can be performed concurrently in terms of data pre-fetch and/or motion vector refinement and motion compensation.
- 10    10. The method according to claim 9, wherein the concurrent set comprises the current set or the spatially neighboring set.
11.    The method according to any preceding claim, wherein the spatially neighboring set comprises a coding unit or sub-coding belonging to a first CTB that  
15 is spatially adjacent, such as a upper row CTB, in decoding to the coding unit or sub-coding of the current set of a second CTB, such as the following row CTB.
12. The method according to any preceding claim, wherein the CTB size is selected to be as small as possible, e.g. in reverse order of preference:  $64 \times 64$ ,  $32 \times 32$ , or  
20  $16 \times 16$ .
13. The method according to any of claims 2-12, wherein the concurrency set comprises all coding units (CU) of a CTB.
- 25     14. The method according to any preceding claim, wherein an error surface as defined herein is used in a cost function.
15. A method for using unrefined motion vector(s) for video data of a neighbor  
30 coding unit (CU) that falls in a preceding pipeline slot to the current coding unit's concurrency set to perform pre-fetch, and use the refined motion vectors of a neighbor CU that does not fall within the same concurrency set as start of refinement motion vector center and using padded samples whenever the refinement requires

5 samples that are not pre-fetched based on a configurable additional sample fetch range around the pre-fetch center.

16. A decoding or encoding apparatus, comprising the decoding or encoding apparatus is configured to perform the method of any of claims 1-15.

10

17. A decoding or encoding apparatus for video data such as in DMRV or PMMVD, comprising the decoding or encoding apparatus is configured to:

- pre-fetch data for a spatially neighboring set which comprises at least one coding unit (CU) or sub-coding unit; and
- 15 - compute, using the pre-fetched data, a refined motion vector for the spatially neighboring set;
- determine, for a current set which comprises at least one coding unit (CU), or sub coding unit, that the refined motion vector of the spatially neighboring set is available, wherein the determining is performed before or in a data pre-fetch stage of
- 20 the current set;
- wherein if a refined motion vector is not available, using unrefined motion vector(s) for video data of a neighbor coding unit (CU) that falls in a preceding pipeline slot to that of the current coding unit's concurrency set;
- compute a refined motion vector for the current set
- 25 - using the refined motion vector of the spatially neighboring set which is available, or
- using the refined motion vectors of a neighbor CU that does not fall within the same concurrency set as start of refinement motion vector center, and using padded samples whenever the refinement samples are not pre-fetched based on a
- 30 configurable additional sample fetch range around the pre-fetch center.

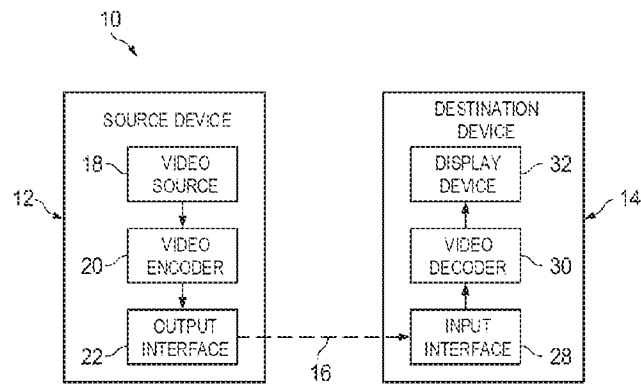


FIG. 1A

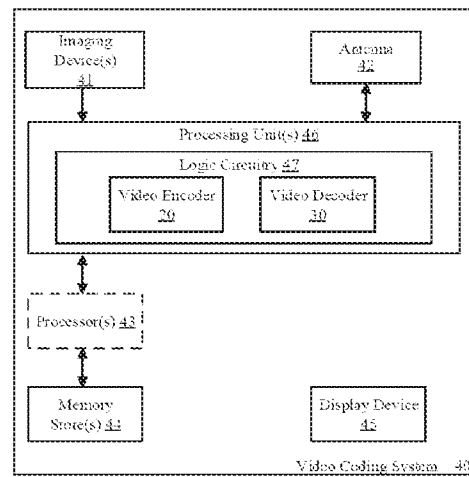


FIG. 1B

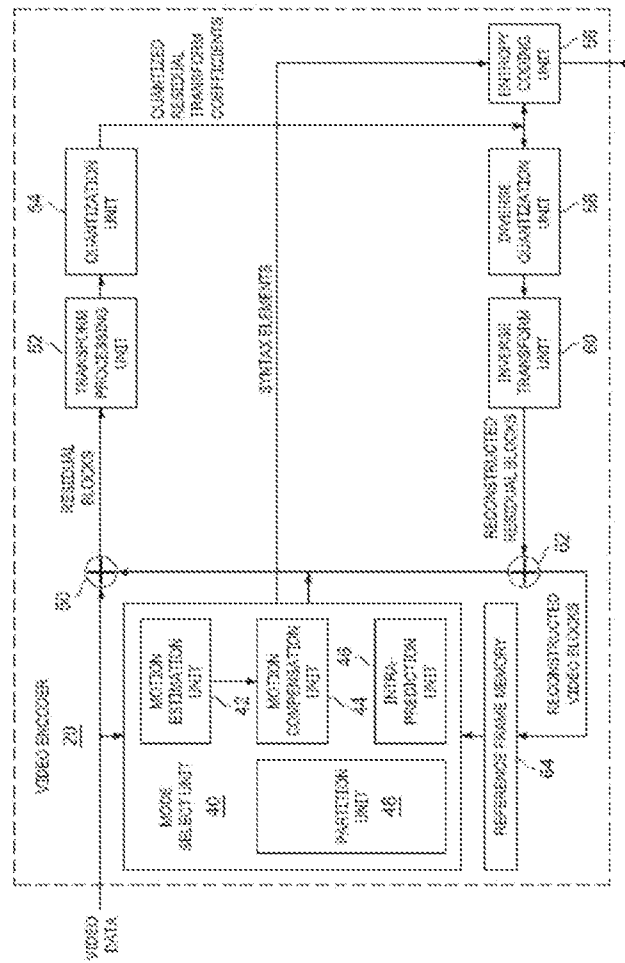


FIG. 2

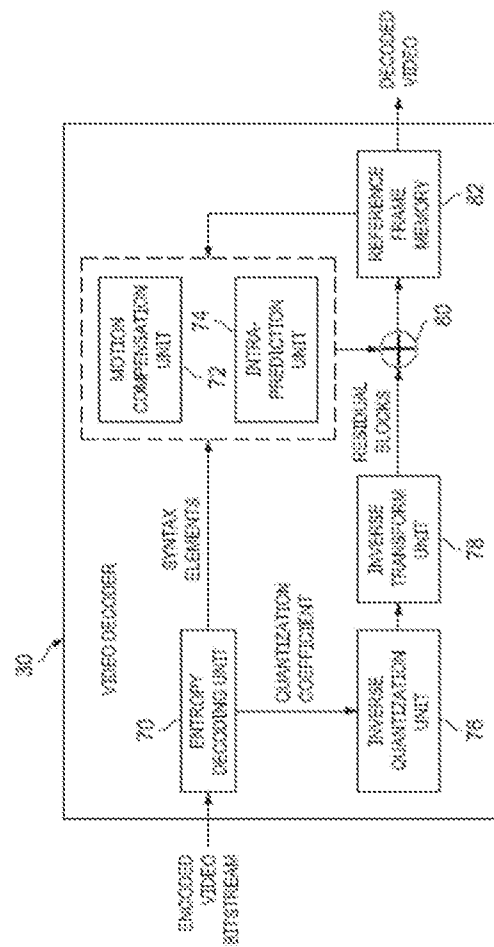


FIG. 3

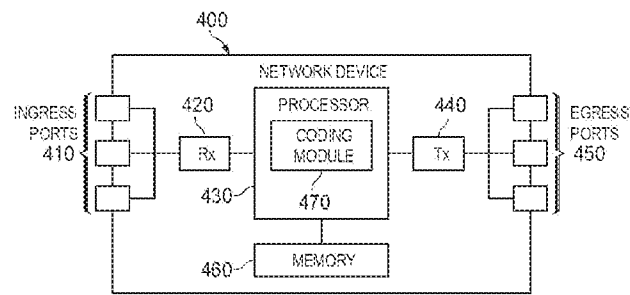


FIG. 4

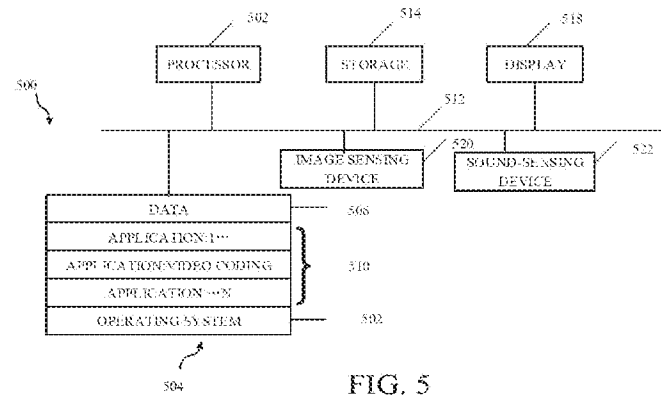


FIG. 5

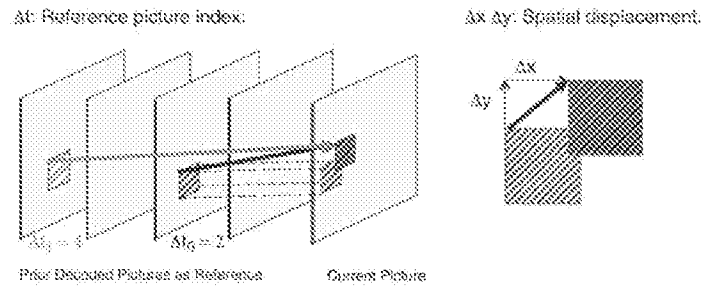


Fig. 6

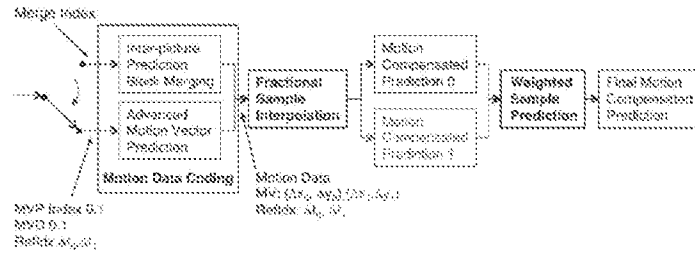


Fig. 7

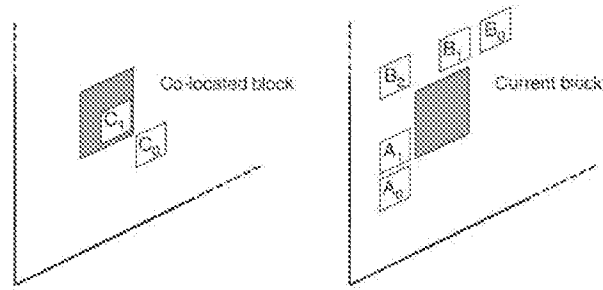


Fig. 8

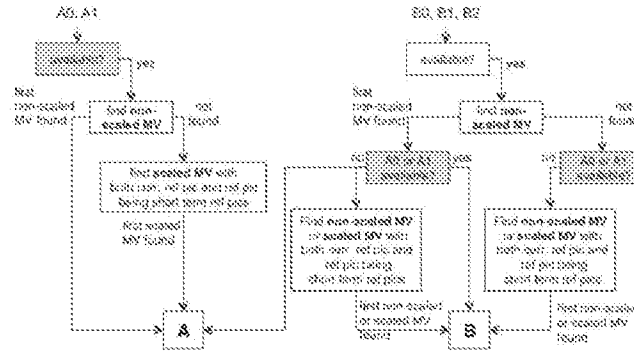


Fig. 9

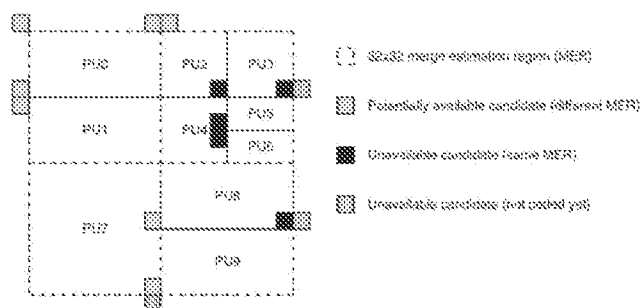


Fig. 10



Fig. 11.

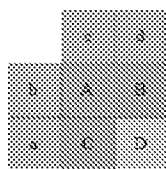


Fig. 12

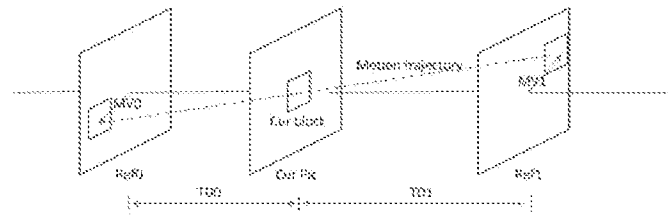


Fig. 13

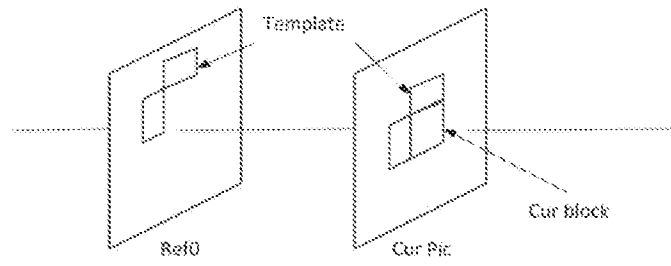


Fig. 14

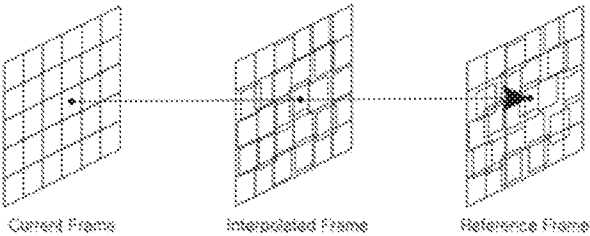


Fig. 15

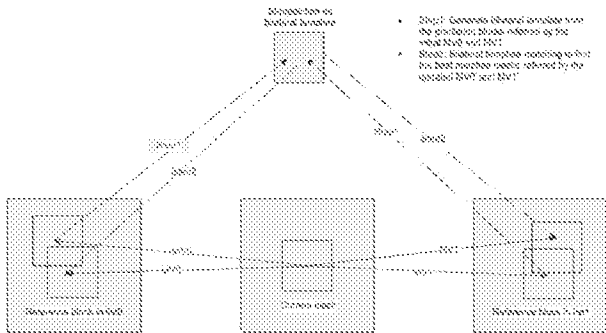


Fig. 16

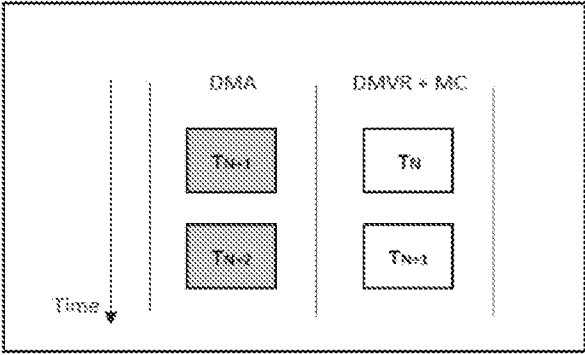


Fig. 17

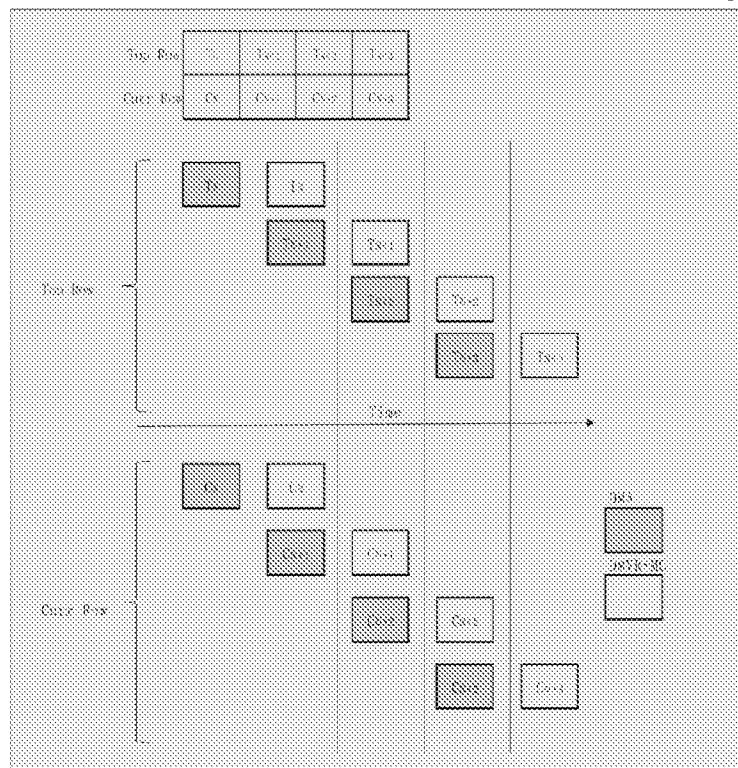


Fig. 13

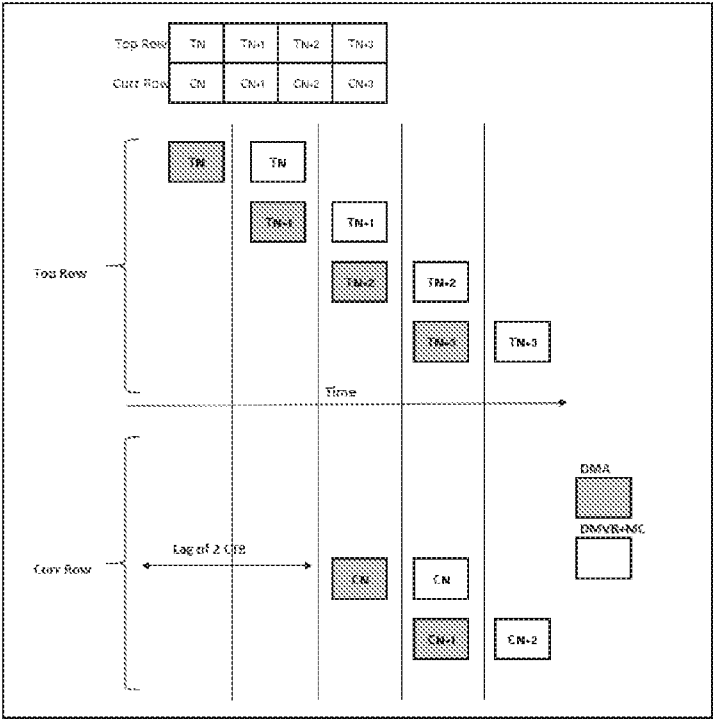


Fig. 19

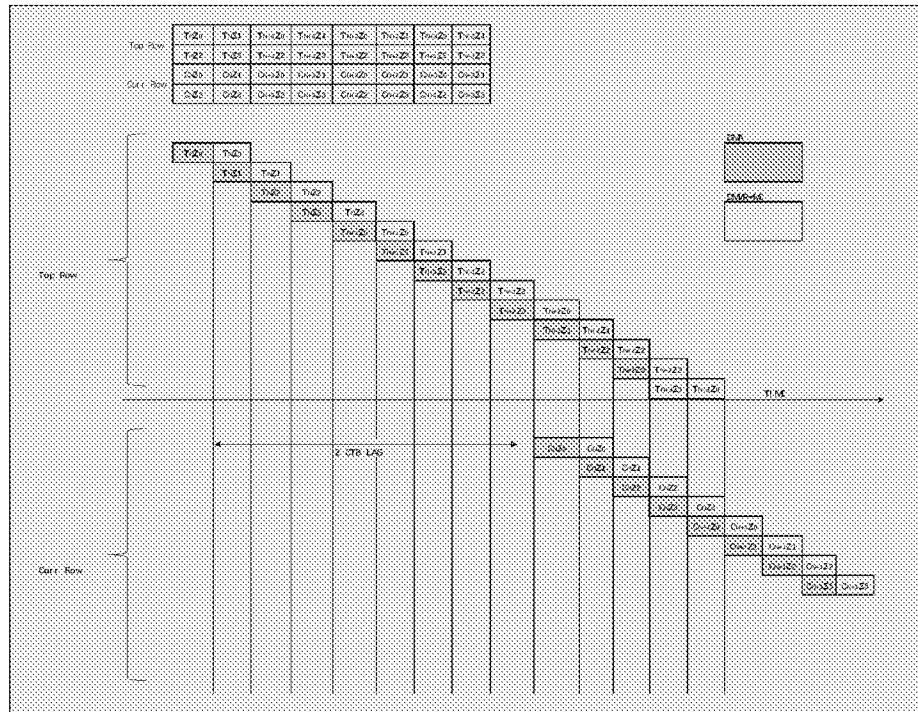


Fig. 20a

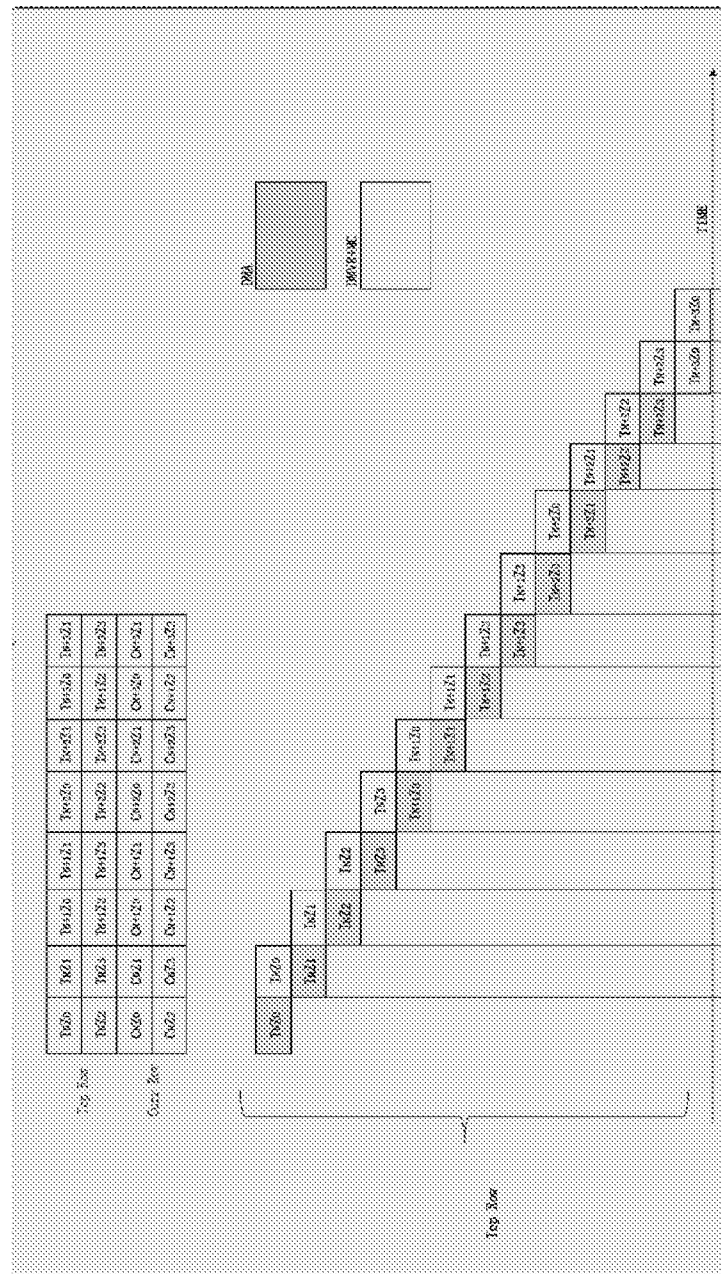


Fig. 20b

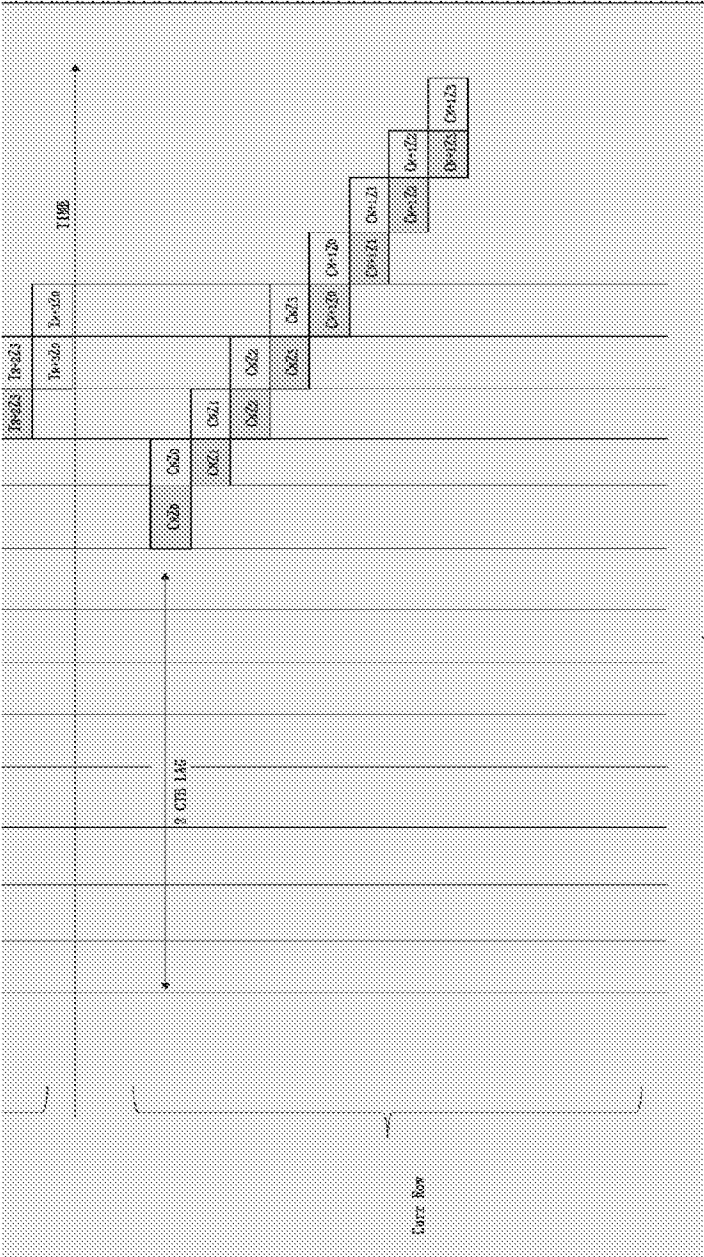


Fig. 20c

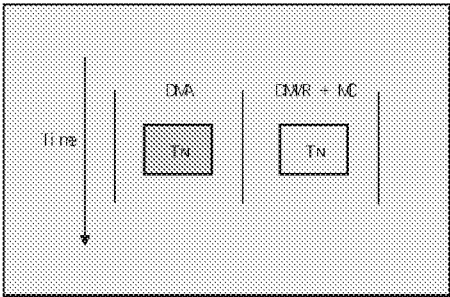


Fig. 21

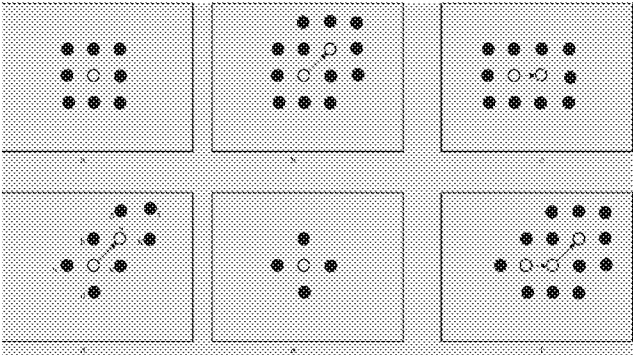


Fig. 22

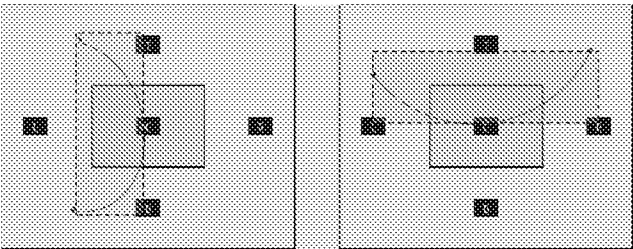


Fig. 23

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2019/094218

**A. CLASSIFICATION OF SUBJECT MATTER**

H04N 19/00(2014.01)i

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

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

H04N

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

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

VEN;WOTXT;EPTXT;USTXT;CNABS;CNKI;CNTXT:refined, motion, vector, decoding, CU, sub, spatially, pipeline, compensation, pre-fetch, stage, CTB, DMA, padded, sample

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages                     | Relevant to claim No. |
|-----------|--------------------------------------------------------------------------------------------------------|-----------------------|
| A         | US 2008031333 A1 (LI XINGHAI BILLY ET AL.) 07 February 2008 (2008-02-07)<br>the whole document         | 1-17                  |
| A         | CN 101325710 A (VIXS SYSTEMS INC) 17 December 2008 (2008-12-17)<br>the whole document                  | 1-17                  |
| A         | US 2012288001 A1 (SUNPLUS TECHNOLOGY CO., LTD.) 15 November 2012<br>(2012-11-15)<br>the whole document | 1-17                  |



Further documents are listed in the continuation of Box C.



See patent family annex.

\* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

09 September 2019

Date of mailing of the international search report

19 September 2019

Name and mailing address of the ISA/CN

National Intellectual Property Administration, PRC  
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing  
100088  
China

Authorized officer

QI,Jingwei

Facsimile No. (86-10)62019451

Telephone No. 86- (010) -62411482

**INTERNATIONAL SEARCH REPORT**  
**Information on patent family members**

International application No.

**PCT/CN2019/094218**

| Patent document<br>cited in search report |            |    | Publication date<br>(day/month/year) | Patent family member(s) |            |    | Publication date<br>(day/month/year) |
|-------------------------------------------|------------|----|--------------------------------------|-------------------------|------------|----|--------------------------------------|
| US                                        | 2008031333 | A1 | 07 February 2008                     | None                    |            |    |                                      |
| CN                                        | 101325710  | A  | 17 December 2008                     | US                      | 8218636    | B2 | 10 July 2012                         |
|                                           |            |    |                                      | EP                      | 1942675    | A2 | 09 July 2008                         |
|                                           |            |    |                                      | US                      | 2008117971 | A1 | 22 May 2008                          |
|                                           |            |    |                                      | EP                      | 1942675    | A3 | 07 March 2012                        |
|                                           |            |    |                                      | CN                      | 101325710  | B  | 15 February 2012                     |
|                                           |            |    |                                      | EP                      | 1942675    | B1 | 26 March 2014                        |
| US                                        | 2012288001 | A1 | 15 November 2012                     | TW                      | 201246920  | A  | 16 November 2012                     |
|                                           |            |    |                                      | US                      | 8761262    | B2 | 24 June 2014                         |