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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 October 2019 (17.10.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/199415 A1

(51) International Patent Classification:
H04N 19/597 (2014.01) *H04N 19/503* (2014.01)

(21) International Application Number:
PCT/US2019/022776

(22) International Filing Date:
18 March 2019 (18.03.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/657,338 13 April 2018 (13.04.2018) US

(71) Applicant: **FUTUREWEI TECHNOLOGIES, INC.** [US/US]; 5700 Tennyson Parkway, Suite 600, Plano, Texas 75024 (US).

(72) Inventors: **ZHANG, Dejun**; Huawei Building No. 3 Xinxi Road, Shangdi Information Industry Base, Haidian District, Beijing (CN). **ZAKHARCHENKO, Vladyslav**; 457 E. Evelyn Avenue, Apt. 147, Sunnyvale, California 94086 (US). **CHEN, Jianle**; 10756 Corte De Tiburon, San Diego, California 92130 (US). **WU, Wenhai**; Huawei Building No. 3 Xinxi Road, Shangdi Information Industry Base, Haidian District, Beijing (CN).

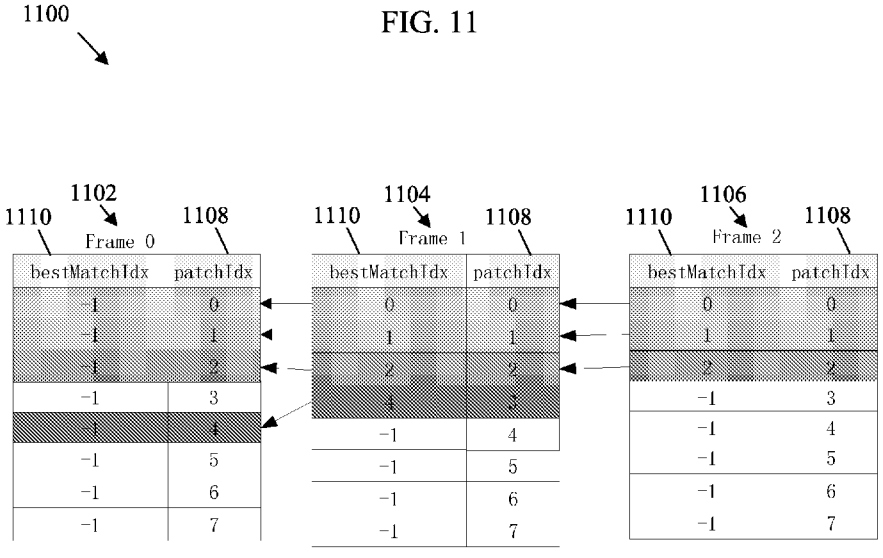
(74) Agent: **HOWELL, Brandt D.** et al.; Conley Rose, P.C., 5601 Granite Parkway, Suite 500, Plano, Texas 75024 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, 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).

(54) Title: DIFFERENTIAL CODING METHOD FOR PATCH SIDE INFORMATION

FIG. 11



(57) Abstract: A method of media coding implemented by a coding device. The method includes generating a patch list identifying matched patches between a current patch frame and a reference patch frame, and encoding a difference between patch side information of a first patch in the current patch frame and patch side information of a second patch in the reference patch frame when a best match patch index value for the first patch in the patch list of the current patch frame matches a second patch index value in a second patch list of the reference patch frame.

[Continued on next page]

WO 2019/199415 A1

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

Differential Coding For Patch Side Information

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims the benefit of U.S. Provisional Patent Application No. 62/657,338 filed April 13, 2018 by Dejun Zhang and titled "Differential Coding Of Patch Side Information," which is hereby incorporated by reference.

TECHNICAL FIELD

[0002] The present disclosure is generally related to point cloud coding, and specifically to a temporally consistent multi-frame patch packing in the context of point cloud coding.

BACKGROUND

[0003] The point cloud is employed in a wide variety of applications including entertainment industry, intelligent automobile navigation, geospatial inspection, three dimensional (3D) modelling of real world objects, visualization etc. Considering the non-uniform sampling geometry of the point cloud, compact representations for storage and transmission of such data is useful. Compared with the other 3D presentations, the irregular point cloud is more general and applicable for a wider range of sensors and data acquisition strategies. For example, when performing a 3D presentation in a virtual reality world or remote renderings in a tele-presence environment, the rendering of virtual figures and real-time instructions are processed as dense point cloud data set.

SUMMARY

[0004] A first aspect relates to a method of media coding implemented by a coding device. The method includes generating, using the coding device, a patch list identifying matched patches between a current patch frame and a reference patch frame; and encoding, using the coding device, a difference between patch side information of a first patch in the current patch frame and patch side information of a second patch in the reference patch frame when a best match patch index value for the first patch in the patch list of the current patch frame matches a second patch index value in a second patch list of the reference patch frame.

[0005] A second aspect relates to a method of media coding implemented by a coding device. The method includes obtaining patch side information for all patches in a current frame; encoding patch parameters corresponding to the patches; iterating a patches buffer

containing a list of the patches, the patches buffer organized based on matches between consecutive frames; encoding a value in a best matches index of the patches buffer for a first patch; determining that there is a match between the current frame and a reference frame using the best matches index; obtaining the patch side information from a reference frame; and encoding a difference between the patch side information for the first patch in the current frame and the patch side information from the reference frame.

[0006] A third aspect relates to a method of media coding implemented by a coding device. The method includes obtaining patch side information for all patches in a current frame; encoding patch parameters corresponding to the patches; iterating a patches buffer containing a list of the patches, the patches buffer organized based on matches between consecutive frames; determining that the patches buffer includes a global matched patch; determining that there is a match between the current frame and a reference frame using the best matches index when the patches buffer includes the global matched patch; obtaining the patch side information from a reference frame; and encoding a difference between the patch side information for the first patch in the current frame and the patch side information from the reference frame when the match has been determined.

[0007] A fourth aspect relates to a method of media coding implemented by a coding device. The method includes obtaining patch side information for all patches in a group of frames; encoding global matched patches for the group of frames; determining whether there are global matched patches for the group of frames; iterate each frame in the group of frames and a patches buffer for the current frame; encode a difference between patch side information for a first patch in the current frame and the patch side information from a reference frame when there are global matched patches; iterate each frame in the group of frames and a patches buffer for the current frame; and encode: a difference between the patch side information for the patch in the current frame and the patch side information for the patch in the reference frame when there is a local matched patch; or patch coordinates for the patch in the current frame, a difference between a size of the patch and a size of a previously encoded patch within the current frame; and a normal axis for a current patch.

[0008] A fifth aspect relates to a method of media coding implemented by a coding device. The method includes obtaining patch side information for all patches in a current frame from an encoded point cloud stream; decoding patch parameters corresponding to the patches; iterating a patches buffer containing a list of the patches, the patches buffer organized based on matches between consecutive frames; decoding a value in a best matches index of the patches buffer for a first patch; determining that there is a match between the first patch in the current

frame and a second patch in the reference frame using the best matches index; obtaining the patch side information from a reference frame; and decoding a difference between the patch side information for the first patch in the current frame and the patch side information for the second patch from the reference frame.

[0009]

[0010] The methods facilitate signaling techniques that take advantage of temporal similarities between patches. That is, a patch location in a texture/geometry/occupancy map may be further refined to ensure optimal sub-block location for temporal-based predictive compression. In addition, a temporal correlation between several patches in frames with a different temporal index may be utilized in a temporally-consistent patch patching method in order to permit delta coding of the patch side information.

[0011] In a first implementation form of the method according to the first, second, third, fourth, or fifth aspect as such, the coding device is a video encoder or a video decoder.

[0012] In a second implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the patch side information includes one or more of a height, a width, and a depth of the patches.

[0013] In a third implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the patch parameters include one or more of a patch count, an occupancy precision, a maximum candidate count, a bit count for an x-coordinate in two dimensions (2D), a bit count for a y-coordinate in 2D, a bit count for an x-coordinate in three dimensions (3D), a bit count for a y-coordinate in 3D, and a bit count for a z-coordinate in 3D.

[0014] In a fourth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the patch count represents a number of patches for an image.

[0015] In a fifth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the occupancy count represents an amount of downsampling used.

[0016] In a sixth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the maximum candidate count represents a number of patches that belong to a same block of the occupancy precision.

[0017] In a seventh implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the bit count represents a number of bits needed to describe a coordinate.

[0018] In an eighth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the patches buffer is organized based on any matches between consecutive frames.

[0019] In a ninth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the global matched patches are stored at a beginning of the patches buffer.

[0020] In a ninth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the local matched patches are stored after the global matched patches in the patches buffer.

[0021] In a tenth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, unmatched patches are stored after the global matched patches and the local matched patches in the patches buffer.

[0022] In an eleventh implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, one or more parameters from the patch side information of the reference frame are inherited for the patch side information of the current frame.

[0023] In a twelfth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, one or more metadata parts from the reference frame are inherited for the current frame.

[0024] In a thirteenth implementation form of the method according to the first, second, third, fourth, or fifth aspect as such or any preceding implementation form of the first through fifth aspects, the one or more metadata parts include a patch rotation, a scale parameter, and a material identifier.

[0025] A sixth aspect relates to a coding apparatus including that includes a receiver configured to receive a picture to encode or to receive a bitstream to decode, a transmitter coupled to the receiver, the transmitter configured to transmit the bitstream to a decoder or to transmit a decoded image to a display, a memory coupled to at least one of the receiver or the

transmitter, the memory configured to store instructions, and a processor coupled to the memory, the processor configured to execute the instructions stored in the memory to perform the method of any of the preceding aspects or implementations.

[0026] The coding apparatus facilitates signaling techniques that take advantage of temporal similarities between patches. That is, a patch location in a texture/geometry/occupancy map may be further refined to ensure optimal sub-block location for temporal-based predictive compression. In addition, a temporal correlation between several patches in frames with a different temporal index may be utilized in a temporally-consistent patch patching method in order to permit delta coding of the patch side information.

[0027] In a first implementation form of the coding apparatus according to the sixth aspect as such, the apparatus further includes a display configured to display an image.

[0028] A seventh aspect relates to a system that includes an encoder and a decoder in communication with the encoder. The encoder or the decoder includes the coding apparatus of any of the preceding aspects or implementations.

[0029] The system facilitates signaling techniques that take advantage of temporal similarities between patches. That is, a patch location in a texture/geometry/occupancy map may be further refined to ensure optimal sub-block location for temporal-based predictive compression. In addition, a temporal correlation between several patches in frames with a different temporal index may be utilized in a temporally-consistent patch patching method in order to permit delta coding of the patch side information.

[0030] An eighth aspect relates to a means for coding that includes receiving means configured to receive a picture to encode or to receive a bitstream to decode, transmission means coupled to the receiving means, the transmission means configured to transmit the bitstream to a decoder or to transmit a decoded image to a display means, storage means coupled to at least one of the receiving means or the transmission means, the storage means configured to store instructions, and processing means coupled to the storage means, the processing means configured to execute the instructions stored in the storage means to perform the methods in any of the preceding aspects or implementations.

[0031] The means for coding facilitates signaling techniques that take advantage of temporal similarities between patches. That is, a patch location in a texture/geometry/occupancy map may be further refined to ensure optimal sub-block location for temporal-based predictive compression. In addition, a temporal correlation between several patches in frames with a different temporal index may be utilized in a temporally-consistent patch patching method in order to permit delta coding of the patch side information.

[0032] The features disclosed herein may be utilized to improve the signaling of virtual reality content having multiple viewpoints in a video bitstream. The improved signaling enhances the performance of virtual reality (VR) video systems, e.g., by indicating or identifying which particular viewpoint from several available viewpoints corresponds to a viewport-related immersive media metric.

[0033] 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.

[0034] These and other features will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0035] For a more complete understanding of this disclosure, reference is now made to the following brief description, taken in connection with the accompanying drawings and detailed description, wherein like reference numerals represent like parts.

[0036] FIG. 1 is a block diagram illustrating an example coding system that may utilize media coding techniques such as point cloud coding.

[0037] FIG. 2 a block diagram illustrating an example video encoder that may implement media coding techniques.

[0038] FIG. 3 a block diagram illustrating an example video decoder that may implement media coding techniques.

[0039] FIG. 4 is an example point cloud media that may be encoded/decoded using the coding system of FIG. 1.

[0040] FIG. 5 is an example of texture segmentation and packing for point cloud media.

[0041] FIG. 6 illustrates a coding method using a patch packing process.

[0042] FIG. 7 is an algorithm that may be used to implement a portion of the method of FIG. 6.

[0043] FIG. 8 illustrates an embodiment of a compression pipeline.

[0044] FIG. 9 illustrates a group of frames (GOF).

[0045] FIG. 10 illustrates an embodiment of a current frame including a representation of patches from several frames.

[0046] FIG. 11 illustrates an embodiment of a patch index buffer flow for a GOF.

[0047] FIG. 12 illustrates an embodiment of a coding method for a patch packing process.

[0048] FIG. 13 illustrates an algorithm that may be used to implement a portion of the method of FIG. 12.

[0049] FIG. 14 an embodiment of a coding method for a patch packing process.

[0050] FIG. 15 an algorithm that may be used to implement a portion of the method of FIG. 14.

[0051] FIG. 16 an embodiment of a coding method for a patch packing process.

[0052] FIG. 17 an algorithm that may be used to implement a portion of the method of FIG. 16.

[0053] FIG. 18 is a schematic diagram of an example media coding device.

[0054] FIG. 19 is a schematic diagram of an embodiment of a means for coding.

[0055] FIG. 20 illustrates an embodiment of a decoding method corresponding to the patch packing process.

DETAILED DESCRIPTION

[0056] 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 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.

[0057] FIG. 1 is a block diagram illustrating an example coding system 10 that may utilize media coding (a.k.a., video coding) techniques such point cloud coding. As shown in FIG. 1, 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-readable medium 16. Source device 12 and destination device 14 may comprise any of a wide range of devices, including desktop computers, notebook (e.g., 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, or the like. In some cases, source device 12 and destination device 14 may be equipped for wireless communication.

[0058] 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 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 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 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.

[0059] 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.

[0060] The techniques of this disclosure are not necessarily limited to wireless applications or settings. The techniques may be applied to media 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 one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

[0061] In the example of FIG. 1, 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 disclosure, video encoder 20 of the source device 12 and/or the video decoder 30 of the destination device 14 may be configured to apply the techniques for media coding. 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 external camera. Likewise, destination device 14 may interface with an external display device, rather than including an integrated display device.

[0062] The illustrated coding system 10 of FIG. 1 is merely one example. Techniques for media coding may be performed by any digital media (or video) encoding and/or decoding device. Although the techniques of this disclosure generally are performed by a media 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.

[0063] Source device 12 and destination device 14 are merely examples of such 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 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.

[0064] 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 computer-generated video.

[0065] 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 media coding in general, and may be applied to wireless and/or 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.

[0066] Computer-readable medium 16 may include transient media, such as a 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 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.

[0067] 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 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.

[0068] 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 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 standards include MPEG-2 and ITU-T H.263. Although not shown

in FIG. 1, 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 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).

[0069] Video encoder 20 and video decoder 30 each may be implemented as any 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 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 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.

[0070] FIG. 2 is a block diagram illustrating an example of video encoder 20 that may implement media coding techniques. 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 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 (a.k.a., uni prediction) prediction (P mode) or bi-prediction (a.k.a., bi prediction) (B mode), may refer to any of several temporal-based coding modes.

[0071] As shown in FIG. 2, video encoder 20 receives a current video block 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 (a.k.a., intra prediction) unit 46, and partition unit 48. For video block 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) 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).

[0072] 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 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.

[0073] 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 units (sub-CUs) based on rate-distortion analysis (e.g., rate-distortion optimization). Mode select unit 40 may further produce a quad-tree data structure indicative of partitioning of a LCU into sub-CUs. Leaf-node CUs of the quad-tree may include one or more prediction units (PUs) and one or more transform units (TUs).

[0074] 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 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 mode encoded, or inter-prediction (a.k.a., 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 quad-tree. A TU can be square or non-square (e.g., rectangular) in shape.

[0075] 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, partition information, and other such syntax information, to entropy coding unit 56.

[0076] 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 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 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 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 fractional pixel positions and output a motion vector with fractional pixel precision.

[0077] 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 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.

[0078] Motion compensation, performed by motion compensation unit 44, may 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 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 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.

[0079] Intra-prediction unit 46 may intra-predict a current block, as an alternative 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 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.

[0080] For example, intra-prediction unit 46 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes. 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-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.

[0081] 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 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 depth blocks of a depth map.

[0082] 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 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), 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.

[0083] Video encoder 20 forms a residual video block by subtracting the prediction data from mode select unit 40 from the original video block being coded. Summer 50 represents the component or components that perform this subtraction operation.

[0084] 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. 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.

[0085] Transform processing unit 52 applies the transform to the residual block, producing a block of residual transform coefficients. The transform may convert 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 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.

[0086] Following quantization, entropy coding unit 56 entropy codes the quantized transform coefficients. For example, entropy coding unit 56 may 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) 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.

[0087] 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. 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 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.

[0088] FIG. 3 is a block diagram illustrating an example of video decoder 30 that may implement media coding techniques. 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. 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 from entropy decoding unit 70.

[0089] 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 the video decoder 30 entropy decodes the bitstream to generate quantized coefficients, 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.

[0090] 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 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 (e.g., 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 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.

[0091] 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 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-prediction status for each inter-coded video block of the slice, and other information to decode the video blocks in the current video slice.

[0092] 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 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.

[0093] Data for a texture image corresponding to a depth map or geometry map may be stored in reference frame memory 82. Motion compensation unit 72 may also be configured to inter-predict depth blocks of a depth map.

[0094] FIG. 4 is an example 400 of point cloud media that may be encoded/decoded using the coding system described above. A point cloud is a set of data points in space. Point clouds are generally produced by 3D scanners, which measure a large number of points on the external surfaces of objects around them. The example 400 of point cloud media in FIG. 4 illustrates three bounding boxes 402, 404, and 406. Each of the bounding boxes 402, 404, and 406 represents a portion or segment of a 3D image from a current frame. While the bounding boxes 402, 404, and 406 in FIG. 4 contain a 3D image of a person, other objects may be included in the bounding boxes in practical applications. Each bounding box 402, 404, and 406 includes an x-axis, a y-axis, and z-axis that indicates a number of pixels occupied by the 3D image in the x, y, and z directions, respectively. For example, the x-axis and the y-axis depict about four-hundred pixels (e.g., from about 0-400 pixels) while the z-axis depicts about one-thousand pixels (e.g., from about 0-1000 pixels).

[0095] Each of the bounding boxes 402, 404, and 406 contains one or more patches 408, which are represented by cubes or boxes in FIG. 4. Each patch 408 contains a portion of the overall object within one of the bounding boxes 402, 404, or 406 and may be described or represented by patch size information. The patch side information may be referred to herein as patch data. The patch side information may include, for example, two-dimensional (2D) and/or three-dimensional (3D) coordinates describing a location of the patch 408 within the bounding

box 402, 404, or 406. The patch side information may also include other parameters as disclosed herein and in practical applications. For example, the patch side information may include parameters such as a normalAxis, which is inherited for current patch side information from the reference patch side information. That is, one or more parameters from the patch side information of the reference frame may be inherited for the patch side information of the current frame. In addition, one or more metadata parts (e.g., patch rotation, a scale parameter, a material identifier, etc.) from the reference frame may be inherited for the current frame. The patches 408 may be interchangeably referred to herein as 3D patches or patch data units. A list of the patches 408 in each bounding box 402, 404, or 406 may be generated and stored in a patches buffer in descending order from the largest patch to the smallest patch.

[0096] FIG. 5 is an example 500 of texture segmentation and packing for point cloud media (e.g., the point cloud media in the example 400 of FIG. 4). The example 500 of texture segmentation and packing for point cloud media in FIG. 5 includes a bounding box 502 corresponding to a current frame from a video sequence. The bounding box 502 of FIG. 5 is 2D as opposed to the bounding boxes 402, 404, and 406 of FIG. 4, which are 3D. As shown, the bounding box 502 of FIG. 5 contains numerous patches 504. The patches 504 may be interchangeably referred to herein as 2D patches or patch data units.

[0097] Collectively, the patches 504 in FIG. 5 are a representation of the image in one of the bounding boxes 402, 404, and 406 from FIG. 4. In FIG. 5, the patches 504 collectively represent the image in bounding box 404 of FIG. 4. As such, the 3D image in the bounding box 404 in FIG. 4 is projected onto the boundary box 502 via the patches 504.

[0098] The portions of the bounding box 502 that do not contain one of the patches 504 are referred to as empty space 506. The empty space 506 may also be referred to as void spaces, empty samples, etc.

[0099] Keeping the above in mind, it should be noted that video-based point cloud compression (PCC) codec solutions are based on the segmentation of 3D point cloud data (e.g., the patches 408 of FIG. 4) into 2D projection patches (e.g., the patches 504 of FIG. 5). Indeed, the coding methodology or process described above may be beneficially implemented for various types of technology such as, for example, immersive six degrees of freedom (6 DoF), dynamic Augmented Reality/Virtual Reality (AR/VR) objects, cultural heritage, Graphic Information Systems (GIS), Computer Aided Design (CAD), autonomous navigation, and so on.

[00100] The location for each patch (e.g., one of the patches 504 of FIG. 5) within the bounding box (e.g., the bounding box 502) is typically determined by the size of the patch

alone. For example, the largest of the patches 504 in FIG. 5 is projected onto the bounding box 502 first starting from top-left corner (0, 0). After the largest of the patches 504 has been projected onto the bounding box 502, the next-largest of the patches 504 is projected onto (a.k.a., filled into) the bounding box 502, and so on until the smallest of the patches 504 has been projected onto the bounding box 502. Again, only the size of each patch 504 is considered with this process. In some cases, patches 504 having a smaller size may occupy the space between larger patches and may end up having a position closer to the top, left corner of the bounding box 502 than the larger patches 504.

[00101] Unfortunately, this patch projecting process described above is only suitable for use with intra-prediction where spatial similarities alone are used for prediction and/or compression.

[00102] FIG. 6 illustrates a coding method 600 corresponding to the patch packing process described above. The coding method 600 may be employed by a coding device (e.g., video encoder 20). In block 602, patch side information is obtained for all patches (e.g., the patches 408 in FIG. 4). The patch side information may include parameters such as position (e.g., x,y coordinates of the patches), height, width, and depth of the patches.

[00103] In block 604, various patch parameters are encoded. The patch parameters include, for example, a patch count (e.g., patchCount), an occupancy precision (e.g., occupancyPrecision), a maximum candidate count (e.g., maxCandidateCount), a bit count for an x-coordinate in 2D (e.g., bitCountU0), a bit count for a y-coordinate in 2D (e.g., bitCountV0), a bit count for an x-coordinate in 3D (e.g., bitCountU1), a bit count for a y-coordinate in 3D (e.g., bitCountV1), and a bit count for a z-coordinate in 3D (e.g., bitCountD1). The patch count represents the number of patches for the image. The occupancy precision represents an amount of downsampling used, and the maximum candidate count represents the number of patches that belong to the same block of occupancy precision. The bit counts represent the number of bits needed to describe each of the various coordinates.

[00104] The various patch parameters are encoded in block 604 using a suitable number of bits for the particular parameter being encoded. For example, for a patch count parameter with a value of 250, 8 bits are needed (e.g., $2^8 = 256$, which is greater than 250). Likewise, for a maximum candidate count parameter with a value of 1,000, 10 bits are needed (e.g., $2^{10} = 1,024$, which is greater than 1,000).

[00105] In block 606, a patches buffer containing a list of the patches (e.g., the patches 408 in FIG. 4) is iterated in order to encode the information for each patch. As noted above,

the patches buffer is organized by patch size (e.g., patches[k], where k=1 to n). Therefore, the iterative process in FIG. 6 starts with the largest of the patches.

[00106] In block 608, various patch coordinates are encoded for the current patch (e.g., patch[k]). The patch coordinates include, for example, an x-coordinate of the patch (e.g., u0), a y-coordinate of the patch (e.g., v0), an x-coordinate of the patch in 3D (e.g., u1), a y-coordinate of the patch in 3D (e.g., v1), and a z-coordinate of the patch in 3D (e.g., d1).

[00107] The various patch coordinates are encoded in block 608 using a suitable number of bits for the particular parameter being encoded. For example, for an x-coordinate with a value of 6, 3 bits are needed (e.g., $2^3 = 8$, which is greater than 6). Likewise, for a z-coordinate of 40, 6 bits are needed (e.g., $2^6 = 64$, which is greater than 40).

[00108] In block 610, various patch size deltas are encoded for the current patch. The patch size deltas include, for example, the difference between the size of the current patch and the size of the previously encoded patch within the current frame. The patch size deltas include, for example, a change in the x-coordinate of the current patch relative to the previous patch (e.g., deltaSizeU0) and a change in the y-coordinate of the current patch relative to the previous patch (e.g., deltaSizeV0). The various patch size deltas are encoded in block 610 using a suitable number of bits for the particular parameter being encoded.

[00109] In block 612, the normal axis is encoded for the current patch. The normal axis represents the side of the 3D bounding box of the patch (e.g., patch 408 in FIG. 4) being projected. For example, if the back side of the patch is being considered, the normal axis points outward from the page in FIG. 4 and the normal axis may have a value of z = -1. If, however, the front side of the patch is being considered, the normal axis points inward into the page in FIG. 4 and the normal axis may have a value of z = 1. The normal axis is encoded in block 612 using a suitable number of bits for the particular parameter being encoded as explained above. As used herein, blocks 608-612 may be collectively referred to as "Anchor's method."

[00110] Following the execution of block 612, the method 600 returns to block 606 to continue the iterative process for the next largest of the patches in the patches buffer. The iterative process continues until the smallest of the patches in the current frame has been considered. By performing the method of FIG. 6, the patches 408 in FIG. 4 are effectively projected into the boundary box 502 of FIG. 5.

[00111] FIG. 7 is an algorithm 700 that may be used to implement a portion of the method 600 of FIG. 6. In particular, the algorithm 700 may be used to implement blocks 604-612 from FIG. 6.

[00112] FIG. 8 illustrates an embodiment of a compression pipeline 800 corresponding to the methods described above. The compression pipeline 800 may be implemented by a coding device (e.g., video encoder 20) using hardware, software, firmware, or some combination thereof. As shown, a point cloud frame is input 802 into a decomposition into patches module 804. The point cloud frame may be similar to the example 400 of point cloud media in FIG. 4. The decomposition into patches module 804 generates a list of patches (e.g., patches 408 in FIG. 4). The list of patches is output 806 to a packing module 808 and an auxiliary patch-information compression module 810. The packing module 808 and auxiliary patch-information compression module 810 may be implemented using hardware, software, firmware, or some combination thereof. In an embodiment, the packing module 808 may comprise a packing processor, a packing device, or a packer at least partially implemented using hardware. In an embodiment, the auxiliary patch-information compression module 810 may comprise an auxiliary patch-information compression processor, compressing device, or a compressor at least partially implemented using hardware.

[00113] The packing module 808 uses the list of patches to generate a 2D bounding box (e.g., the bounding box 502) and an occupancy map. The occupancy map indicates which areas of the bounding box are occupied by patches (e.g., patches 504) and which areas of the bounding box are occupied by empty space (e.g., empty space 506). The 2D bounding box is output 812 to the geometry generation unit 814, the texture generation unit 816, and the attribute unit 818. The geometry generation unit 814, the texture generation unit 816, and the attribute unit 818 generate the geometry, a texture, and attributes for an image and output 820 those parameters to the video compression unit 822. The occupancy map is output 824 to the occupancy map compression module 826.

[00114] The video compression module 822 compresses the image generated using the geometry, a texture, and attributes to generate a compressed image. The compressed image is output 828 to a multiplexer 830. The occupancy map compression module 826 compresses the occupancy map to generate a compressed occupancy map. The compressed occupancy map is output 832 to the multiplexer 830. The auxiliary patch-information compression module 810 compresses the list of patches to generate a compressed list of patches. This process may be accomplished, for example, using the algorithm 700 in FIG. 7. The compressed list of patches is output 834 to the multiplexer 830. The multiplexer 830 multiplexes the compressed image, the compressed occupancy map, and the compressed list of patches to generate a compressed bitstream, which is output 836 from the multiplexer 830.

[00115] One drawback of the existing method for video-based point cloud compression as described above is that temporal consistency is broken when only patch size is considered during texture image construction (e.g., projecting the patches 408 of FIG. 4 into the boundary box 502 as patches 504). In addition, patches change during the video sequence. For example, a patch may change in size from a reference frame to a current frame. Unfortunately, current video encoders are unable to suitably account for these changes. As such, a bitrate must be increased for the video sequence, which is inefficient.

[00116] Disclosed herein is a method of patch packing that overcomes the aforementioned problems. As will be more fully explained below, improvements to the packing module 808 and the auxiliary patch-information 810 module of FIG. 8 are made. Therefore, a patch location in a texture/geometry/occupancy map may be further refined to ensure optimal sub-block location for temporal-based predictive compression. In addition a temporal correlation between several patches in frames with a different temporal index may be utilized in a temporally-consistent patch patching method in order to permit delta coding of the patch side information.

[00117] FIG. 9 illustrates a group of frames (GOF) 900 that includes a first reference frame 902, a second reference frame 904, a third reference frame 906, and a current frame 908. The arrow 910 represents the progression of frames relative to time, *t*. The group of frames 900 comprising frames 902-908 may be referred to herein as consecutive frames or a temporal sequence of frames. Although three of the reference frames 902-906 are shown in FIG. 9, it should be appreciated that any number of reference frames may be utilized in the GOF 900 in practical applications.

[00118] FIG. 10 illustrates an embodiment of a current frame 1000 including a representation of patches from several frames. The current frame 1000 is similar to the current frame 908 in FIG. 9. As shown, the current frame 1000 includes a representation of a first patch 1002 and a representation of a second patch 1004. The representation of the first patch 1002 corresponds to a patch (e.g., one of the patches 504) from a frame (e.g., frame 902) at time 1. The representation of the second patch 1004 corresponds to the same patch except from a frame (e.g., frame 904) at time 2.

[00119] Assuming that the representation of the first patch 1002 corresponds to the minimum location (e.g., min(x,y|1..k) of the patch over time and the representation of the second patch 1004 corresponds to the maximum location of the patch over time (e.g., max(x,y|1..k) relative to the top, left corner of the current frame 1000, a temporally consistent boundary box 1006 may be established. The temporally consistent boundary box 1006

captures the extent of the location change for the patch over a series of frames (e.g., frames 900-908).

[00120] In an embodiment, the packing module 808 of FIG. 8 generates a list of patches (e.g., patches 504) in a group of frames (e.g., GOF 900) similarly located within their respective frame and, therefore, fall within the boundaries of a temporally consistent boundary box (e.g., the temporally consistent boundary box 1006). The patches that fall within the boundary of the temporally consistent boundary box may be referred to as matched patches, while any patches that fall outside the boundary of the temporally consistent boundary box may be referred to as unmatched patches. As used herein, the list of patches generated by the packing module may be referred to as a patch index.

[00121] FIG. 11 illustrates an embodiment of a patch index buffer flow 1100 for a GOF. The patch index buffer flow 1100 depicts patch frame data unit 1102 (e.g., Frame 0), patch frame data unit 1104 (e.g., Frame 1), and patch frame data unit 1106 (e.g., frame 2). While three of the patch frame data units 1102-1106 are shown in FIG. 11, any number of patch frame data units may be included in the GOF in practical applications. The patch frame data units 1102-1106 may be individually referred to as a patch list and collectively referred to as a frame buffer. Each of the patch frame data units 1102-1106 may be referred to herein as a current patch frame or a reference patch frame. For example, the patch frame data unit 1104 may be referred to as a current patch frame, and the patch frame data units 1102, 1106 may be referred to as reference patch frames.

[00122] Each of the patch frame data units 1102-1106 includes a patch index 1108 (e.g., patchIdx) and a best match index 1110 (e.g., bestMatchIdx or RefPatchIdx). The patch index 1108 in each patch frame data unit 1102-1106 contains a list of patches in numerical order (e.g., from 0 to 7). The best match index 1110 includes a best match index value corresponding to a particular patch. For example, in patch frame data unit 1104 the patch with patch index 3 has a best match index value of 3. As depicted in FIG. 11 and as will be more fully explained below, the best match index 1110 identifies the matched patches between a current patch frame (e.g., patch frame data unit 1104) and a reference patch frame (e.g., patch frame data unit 1102 or patch frame data unit 1106).

[00123] The patches corresponding to patch index 0, 1, and 2 are referred to as global matched patches since they are present in all of the patch frame data units 1102-1106. The patch corresponding to patch index 3 in patch frame data unit 1104 and patch index 4 in patch frame data unit 1102 is referred to a local match patch since that patch is present in some, but not all, of the patch frame data units (e.g., in patch frame data units 1102-1104, but not 1106).

The patches having a best match index value of -1 are referred to as unmatched patches. In an embodiment, the global matched patches are stored at the beginning of each patch frame data unit 1102-1106, followed by the local matched patches, and then the unmatched patches. In an embodiment, a patch coding mode of P_INTER patch type is utilized to identify global and local matched patches. In an embodiment, a patch coding mode of I_INTRA or P_INTRA type is utilized to identify unmatched patches.

[00124] Still referring to FIG. 11, when frame 1104 is being encoded, the patch having patch index 0 is considered first. The best match index for the patch having patch index 0 in frame 1104 has a value of 0. Therefore, instead of encoding all the information for the patch with having patch index 0 in frame 1104, only a differential between the patch side information for the patch having index 0 in frame 1102 (e.g., the reference frame) and the patch side information for the patch having the patch index of 0 in frame 1104 is encoded.

[00125] Likewise, when frame 1104 is being encoded, the patch having patch index 3 is considered. The best match index for the patch having patch index 3 in frame 1104 has a value of 4. Therefore, instead of encoding all the information for the patch having patch index 3 in frame 1104, only a differential between the patch side information for the patch having index 4 in frame 1102 (e.g., the reference frame) and the patch side information for the patch having the patch index of 3 in frame 1104 is encoded.

[00126] As will be more fully explained below, the differential coding scheme described above is utilized to reduce redundancy in the transmission and storage of patch size information. For example, instead of using a direct patch size coding for each patch, a difference between similar patches is transmitted using the global matched patches and the local matched patches as shown in FIG. 11. In an embodiment, the differential coding scheme is performed at least in part by the auxiliary patch-information compression module 810 in FIG. 8.

[00127] Table 1 below depicts patch side information that may be encoded to facilitate the differential coding scheme. The table includes, among other parameters, a numGlobalMatchedPatches value representing the number of global matched patches in the patch frame data units (e.g., patch frame data units 1102-1106) and a bestMatchedIndex value representing the best matched index in a reference frame (e.g., a previous frame).

Table 1 – Patch side information

Patch side information	Description
numGlobalMatchedPatches	Number of matched patches in a coding GoF
patchCount	Total number of patches in a single picture/frame

bestMatchedIndex	The best matched patch index in previous frames, if bestMatchedIndex equals to -1, it means that not find the matched patch in previous frame.
patchIdx	The index of the patch in the frame
occupancyPrecision	The precision of occupancy map
maxCandidateCount	The maximum number of possible overlapping patches in a block(etc: 16*16)
u0	x-coordinate of top-left corner of the patch in occupancy map
v0	y-coordinate of top-left corner of the patch in occupancy map
u1	Tangential shift of the patch in 3d space. The minimum 3d location x of the patch in 3d space
v1	Bitangential shift of the patch in 3d space. The minimum 3d location y of the patch in 3d space
d1	Depth shift of the patch in 3d space. The minimum 3d location z of the patch in 3d space
sizeU0	The width of the patch in occupancy map
sizeV0	The height of the patch in occupancy map
bitCountU0	The number of bits for coding u0
bitCountV0	The number of bits for coding v0
bitCountU1	The number of bits for coding u1
bitCountV1	The number of bits for coding v1
bitCountD1	The number of bits for coding d1
normalAxis	The projection plane

[00128] In an embodiment, the parameter u0 may be referred to as pdu_2d_shift_u, the parameter v0 may be referred to as pdu_2d_shift_v, the parameter u1 may be referred to as pdu_3d_shift_tangent_axis, the parameter referred to as v1 may be referred to as pdu_3d_shift_bitangent_axis, the parameter referred to as d1 may be referred to as pdu_3d_shift_normal_axis, the parameter referred to as sizeU0 may be referred to as pdu_2d_delta_size_u for intra coded patches or dpdu_2d_delta_size_u for inter coded patches, the parameter referred to as sizeV0 may be referred to as pdu_2d_delta_size_v for intra coded patches or dpdu_2d_delta_size_v for inter coded patches, the parameter referred to as bitCountU0 may be referred to as pdu_2d_shift_u_bit_count_minus1 , the parameter referred to as bitCountV0 may be referred to as pdu_2d_shift_v_bit_count_minus1, the parameter referred to as bitCountU1 may be referred to as pdu_3d_shift_tangent_axis_bit_count_minus1, the parameter referred to as bitCountV1 may be referred to as pdu_3d_shift_bitangent_axis_bit_count_minus1, the parameter referred to as bitCountD1 may be referred to as pdu_3d_shift_normal_axis_bit_count_minus1.

[00129] FIG. 12 illustrates an embodiment of a coding method 1200 corresponding to the patch packing process described above. The coding method 1200 may be employed by a

coding device (e.g., video encoder 20). In an embodiment, the coding method 1200 is performed for the encoding of one frame in a temporally consistent patch packing (TCPP) context.

[00130] In block 1202, patch side information is obtained for all patches (e.g., the patches 408 in FIG. 4). The patch side information may include parameters such as height, width, and depth of the patches.

[00131] In block 1204, various patch parameters corresponding to the patches are encoded. The patch parameters include, for example, a patch count (e.g., patchCount), an occupancy precision (e.g., occupancyPrecision), a maximum candidate count (e.g., maxCandidateCount), a bit count for an x-coordinate in 2D (e.g., bitCountU0), a bit count for a y-coordinate in 2D (e.g., bitCountV0), a bit count for an x-coordinate in 3D (e.g., bitCountU1), a bit count for a y-coordinate in 3D (e.g., bitCountV1), and a bit count for a z-coordinate in 3D (e.g., bitCountD1). The patch count represents the number of patches for the image. The occupancy count represents an amount of downsampling used, and the maximum candidate count represents the number of patches that belong to the same block of occupancy precision. The bit count represents the number of bits needed to describe each of the various coordinates.

[00132] The various patch parameters are encoded in block 1204 using a suitable number of bits for the particular parameter being encoded. For example, for a patch count parameter with a value of 250, 8 bits are needed (e.g., $2^8 = 256$, which is greater than 250). Likewise, for a maximum candidate count parameter with a value of 1,000, 10 bits are needed (e.g., $2^{10} = 1,024$, which is greater than 1,000).

[00133] In block 1206, a patches buffer containing a list of the patches (e.g., the patches 408 in FIG. 4) is iterated in order to encode the information for each patch. Unlike the patches buffer described above for FIG. 6, the patches buffer is organized based on and by considering any matches between consecutive frames. In an embodiment, the global matched patches are stored at the beginning of each patches buffer, followed by the local matched patches, and then the unmatched patches. Therefore, the iterative process in FIG. 12 starts with the first of the matched patches (e.g., a local matched patch).

[00134] In block 1208, the value in the best matches index (e.g., best matches index 1110 in FIG. 11) is encoded for the first patch in the patch frame data unit of the patches buffer. For example, for the patch having the patch index of 0 in FIG. 11 for frame 1104 (e.g., Frame 1), the value of 0 in the best match index is encoded.

[00135] In block 1210, a determination is made whether there are any matches between the current frame and the reference frame. In an embodiment, block 1210 checks whether the value of the best match index that was encoded is equal to -1 (e.g., a default value). When a match exists (e.g., the value of the best match index does not equal -1), then the method 1200 proceeds to block 1212. In block 1212, the patch side information from a reference frame (e.g., frame 1102 in FIG. 11) is obtained.

[00136] In block 1214, a difference between the patch side information for the patch in the current frame and the patch side information for the patch in the reference frame is encoded. For example, a differential value is encoded for various patch coordinates such as an x-coordinate of the patches (e.g., u0), a y-coordinate of the patches (e.g., v0), an x-coordinate of the patches in 3D (e.g., u1), a y-coordinate of the patches in 3D (e.g., v1), and a z-coordinate of the patches in 3D (e.g., d1). In addition, a change in the x-coordinate of the current patch relative to the previous patch (e.g., deltaSizeU0) from the reference frame and a change in the y-coordinate of the current patch relative to the previous patch (e.g., deltaSizeV0) in the reference frame is encoded.

[00137] Following the execution of block 1214, the method 1200 returns to block 1206 to continue the iterative process for the next patches in the patches buffer.

[00138] Referring back to block 1210, when a match does not exist (e.g., the value of the best match index equals -1), then the method 1200 proceeds to block 1216. In block 1216, Anchor's method is performed (e.g., blocks 608-612 in FIG. 6) as described above.

[00139] FIG. 13 is an algorithm 1300 that may be used to implement a portion of the method 1200 of FIG. 12. In particular, the algorithm 1300 may be used to implement blocks 1204-1216 from FIG. 12.

[00140] FIG. 14 illustrates an embodiment of a coding method 1400 corresponding to the patch packing process described above. The coding method 1400 may be employed by a coding device (e.g., video encoder 20). In an embodiment, the coding method 1400 is performed based on the assumption that any global matched patches are stored at the beginning of the patches buffer in each frame followed by any other patches in the TCPP context.

[00141] In block 1402, patch side information is obtained for all patches (e.g., the patches 408 in FIG. 4). The patch side information may include parameters such as height, width, and depth of the patches.

[00142] In block 1404, various patch parameters corresponding to the patches are encoded. The patch parameters include, for example, a patch count (e.g., patchCount), an occupancy precision (e.g., occupancyPrecision), a maximum candidate count (e.g.,

maxCandidateCount), a bit count for an x-coordinate in 2D (e.g., bitCountU0), a bit count for a y-coordinate in 2D (e.g., bitCountV0), a bit count for an x-coordinate in 3D (e.g., bitCountU1), a bit count for a y-coordinate in 3D (e.g., bitCountV1), and a bit count for a z-coordinate in 3D (e.g., bitCountD1). The patch count represents the number of patches for the image. The occupancy count represents an amount of downsampling used, and the maximum candidate count represents the number of patches that belong to the same block of occupancy precision. The bit counts represent the number of bits needed to describe each of the various coordinates.

[00143] The various patch parameters are encoded in block 1404 using a suitable number of bits for the particular parameter being encoded. For example, for a patch count parameter with a value of 250, 8 bits are needed (e.g., $2^8 = 256$, which is greater than 250). Likewise, for a maximum candidate count parameter with a value of 1,000, 10 bits are needed (e.g., $2^{10} = 1,024$, which is greater than 1,000).

[00144] In block 1406, a patches buffer containing a list of the patches (e.g., the patches 408 in FIG. 4) is iterated in order to encode the information for each patch. Unlike the patches buffer described above for FIG. 6, the patches buffer is organized by considering any matches between consecutive frames. In an embodiment, the global matched patches are stored at the beginning of each patches buffer, followed by the local matched patches, and then the unmatched patches. Therefore, the iterative process in FIG. 14 starts with the first of the matched patches (e.g., a global matched patch).

[00145] In block 1407, a determination of whether there are global matched patches is made. When there are global matched patches, the method 1400 proceeds to block 1408. In block 1408, the value in the best matches index (e.g., best matches index 1110 in FIG. 11) is encoded for the first patch in the patches buffer. For example, for the patch having the patch index of 0 in FIG. 11 for frame 1104 (e.g., Frame 1), the value of 0 in the best match index is encoded. When there are not any global matched patches, the method 1400 skips block 1408 and proceeds directly to block 1410.

[00146] In block 1410, a determination is made whether there are any matches between the current frame and the reference frame. In an embodiment, block 1410 checks whether the value of the best match index that was encoded is equal to -1 (e.g., a default value). When a match exists (e.g., the value of the best match index does not equal -1), then the method 1400 proceeds to block 1412. In block 1412, the patch side information from a reference frame (e.g., frame 1102 in FIG. 11) is obtained.

[00147] In block 1414, a difference between the patch side information for the patch in the current frame and the patch side information for the patch in the reference frame is encoded. For example, a differential value is encoded for various patch coordinates such as an x-coordinate of the patches (e.g., u0), a y-coordinate of the patches (e.g., v0), an x-coordinate of the patches in 3D (e.g., u1), a y-coordinate of the patches in 3D (e.g., v1), and a z-coordinate of the patches in 3D (e.g., d1). In addition, a change in the x-coordinate of the current patch relative to the previous patch (e.g., deltaSizeU0) from the reference frame and a change in the y-coordinate of the current patch relative to the previous patch (e.g., deltaSizeV0) in the reference frame in encoded.

[00148] Following the execution of block 1414, the method 1400 returns to block 1406 to continue the iterative process for the next patches in the patches buffer.

[00149] Referring back to block 1410, when a match does not exist (e.g., the value of the best match index equals -1), then the method 1400 proceeds to block 1416. In block 1416, Anchor's method is performed (e.g., blocks 608-612 in FIG. 6) as described above.

[00150] FIG. 15 is an algorithm 1500 that may be used to implement a portion of the method 1400 of FIG. 14. In particular, the algorithm 1500 may be used to implement blocks 1404-1416 from FIG. 14.

[00151] FIG. 16 illustrates an embodiment of a coding method 1600 corresponding to the patch packing process described above. The coding method 1600 may be employed by a coding device (e.g., video encoder 20). In an embodiment, the coding method 1600 is performed for the encoding of multiple frames in the TCPP context.

[00152] In block 1602, patch side information is obtained for all patches (e.g., the patches 408 in FIG. 4) for each frame in a group of frames. The patch side information may include parameters such as height, width, and depth of the patches. In block 1604, the number of global matched patches, if any, is encoded. Using FIG. 11 as an example, the number of global matched patches would be encoded as three. As described above, the number three is encoded using a suitable number of bits. In some cases, the number of global matched patches is encoded as zero when there are no global matched patches present in the group of frames.

[00153] In block 1606, a determination is made whether there are any global matched patches for the group of frames. When there are global matched patches, the method 1600 proceeds to block 1608. In block 1608, each frame in the group of frames with global matched patches is iterated. That is, a first frame in the group of frames is considered, then a

second frame, then a third frame, and so on until all the frames with global matched patches have been considered.

[00154] Starting with a first frame from the group of frames as an example, the method 1600 proceeds to block 1610. In block 1610, a patches buffer (a.k.a., a patch frame data unit 1102-1106) containing a list of the patches (e.g., the patches 408 in FIG. 4) is iterated for a frame (e.g., the current frame). Thereafter, in block 1612, the difference between the patch side information for one frame (e.g., the current frame) and the patch side information of a reference frame is encoded. The method 1600 then loops back to block 1606. When there are additional global matched patches in the group of frames, the method 1600 utilizes blocks 1608, 1610, and 1612 again for each frame as shown in FIG. 16. It should be appreciated that this iterative process is similar to one or more of blocks 1406-1414 in method 1400 in FIG. 14.

[00155] When there are no global matched patches determined at block 1606, the method 1600 proceeds to block 1614. In block 1614, each frame in the group of frames is iterated. That is, a first frame in the group of frames is considered, then a second frame, then a third frame, and so on until all the frames without global matched patches have been considered.

[00156] Starting with a first frame from the group of frames as an example, in block 1616 a patches buffer (a.k.a., a patch frame data unit 1102-1106) containing a list of the patches (e.g., the patches 408 in FIG. 4) is iterated for a frame (e.g., the current frame). Thereafter, in block 1618, one of two different processes is performed. When there are no matches between the current frame and the reference frame (e.g., block 1210 in FIG. 12), then Anchor's method (e.g., block 1216 in FIG. 12) is performed. On the other hand, when there are matches between the current frame and the reference frame, the patch side information from a reference frame (e.g., frame 1102 in FIG. 11) is obtained and a difference between the patch side information for the patch in the current frame and the patch side information for the patch in the reference frame is encoded (e.g., blocks 1212 and 1214 in FIG. 12).

[00157] FIG. 17 is an algorithm 1700 that may be used to implement a portion of the method 1600 of FIG. 16. In particular, the algorithm 1700 may be used to implement blocks 1604-1618 from FIG. 16.

[00158] FIG. 18 is a schematic diagram of a media coding device 1800 (e.g., a video encoder 20, a video decoder 30) according to an embodiment of the disclosure. The media coding device 1800 is suitable for implementing the methods and processes disclosed herein. The media coding device 1800 comprises ingress ports 1810 and receiver units (Rx) 1820 for receiving data; a processor, logic unit, or central processing unit (CPU) 1830 to process the data; transmitter units (Tx) 1840 and egress ports 1850 for transmitting the data; and a memory

1860 for storing the data. The coding device 1800 may also comprise optical-to-electrical (OE) components and electrical-to-optical (EO) components coupled to the ingress ports 1810, the receiver units 1820, the transmitter units 1840, and the egress ports 1850 for egress or ingress of optical or electrical signals.

[00159] The processor 1830 is implemented by hardware and software. The processor 1830 may be implemented as one or more CPU chips, cores (e.g., as a multi-core processor), field-programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), and digital signal processors (DSPs). The processor 1830 is in communication with the ingress ports 1810, receiver units 1820, transmitter units 1840, egress ports 1850, and memory 1860. The processor 1830 comprises a coding module 1870. The coding module 1870 implements the disclosed embodiments described above. The inclusion of the coding module 1870 therefore provides a substantial improvement to the functionality of the coding device 1800 and effects a transformation of the coding device 1800 to a different state. Alternatively, the coding module 1870 is implemented as instructions stored in the memory 1860 and executed by the processor 1830.

[00160] The media coding device 1800 may also include input and/or output (I/O) devices 1880 for communicating data to and from a user. The I/O devices 1880 may include output devices such as a display for displaying video data, speakers for outputting audio data, etc. The I/O devices 1880 may also include input devices, such as a keyboard, mouse, trackball, etc., and/or corresponding interfaces for interacting with such output devices.

[00161] The memory 1860 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 1860 may be volatile and non-volatile and may be read-only memory (ROM), random-access memory (RAM), ternary content-addressable memory (TCAM), and static random-access memory (SRAM).

[00162] FIG. 19 is a schematic diagram of an embodiment of a means for coding 1900. In embodiment, the means for coding 1900 is implemented in a media coding device 1902 (e.g., an encoder 20 or a decoder 30). The media coding device 1902 includes receiving means 1901. The receiving means 1901 is configured to receive a picture to encode or to receive a bitstream to decode. The media coding device 1902 includes transmission means 1907 coupled to the receiving means 1901. The transmission means 1907 is configured to transmit the bitstream to a decoder or to transmit a decoded image to a display means (e.g., one of the I/O devices 1880).

[00163] The media coding device 1902 includes a storage means 1903. The storage means 1903 is coupled to at least one of the receiving means 1901 or the transmission means 1907. The storage means 1903 is configured to store instructions. The media coding device 1902 also includes processing means 1905. The processing means 1905 is coupled to the storage means 1903. The processing means 1905 is configured to execute the instructions stored in the storage means 1903 to perform the methods disclosed herein.

[00164] FIG. 20 illustrates an embodiment of a decoding method 2000 corresponding to the patch packing process described above. The decoding method 2000 may be employed by a decoding device (e.g., video decoder 30). In an embodiment, the decoding method 2000 is performed for the decoding of one frame in a temporally consistent patch packing (TCPP) context.

[00165] In block 2002, patch side information is obtained for all patches (e.g., the patches 408 in FIG. 4) from an encoded point cloud stream. The patch side information may include parameters such as height, width, and depth of the patches.

[00166] In block 2004, various patch parameters corresponding to the patches are decoded. The patch parameters include, for example, a patch count (e.g., patchCount), an occupancy precision (e.g., occupancyPrecision), a maximum candidate count (e.g., maxCandidateCount), a bit count for an x-coordinate in 2D (e.g., bitCountU0), a bit count for a y-coordinate in 2D (e.g., bitCountV0), a bit count for an x-coordinate in 3D (e.g., bitCountU1), a bit count for a y-coordinate in 3D (e.g., bitCountV1), and a bit count for a z-coordinate in 3D (e.g., bitCountD1). The patch count represents the number of patches for the image. The occupancy count represents an amount of downsampling used, and the maximum candidate count represents the number of patches that belong to the same block of occupancy precision. The bit count represents the number of bits needed to describe each of the various coordinates.

[00167] The various patch parameters are decoded in block 2004 using a suitable number of bits for the particular parameter being encoded. For example, for a patch count parameter with a value of 250, 8 bits are needed (e.g., $2^8 = 256$, which is greater than 250). Likewise, for a maximum candidate count parameter with a value of 1,000, 10 bits are needed (e.g., $2^{10} = 1,024$, which is greater than 1,000).

[00168] In block 2006, a patches buffer (a.k.a., a patch frame data unit 1102-1106) containing a list of the patches (e.g., the patches 408 in FIG. 4) is iterated in order to decode the information for each patch. The patches buffer is organized (e.g., following the process in block 1206) based on and by considering any matches between consecutive frames. In an

embodiment, the global matched patches are stored at the beginning of each patches buffer, followed by the local matched patches, and then the unmatched patches. Therefore, the iterative process in FIG. 20 starts with the first of the matched patches (e.g., a local matched patch).

[00169] In block 2008, the value in the best matches index (e.g., best matches index 1110 in FIG. 11) is decoded for the first patch in the patches buffer. For example, for the patch having the patch index of 0 in FIG. 11 for frame 1104 (e.g., Frame 1), the value of 0 in the best match index is decoded.

[00170] In block 2010, a determination is made whether there are any matches between the current frame and the reference frame. In an embodiment, block 2010 checks whether the value of the best match index that was encoded is equal to -1 (e.g., a default value). When a match exists (e.g., the value of the best match index does not equal -1), then the method 2000 proceeds to block 2012. In block 2012, the patch side information from a reference frame (e.g., frame 1102 in FIG. 11) is obtained.

[00171] In block 2014, a difference between the patch side information for the patch in the current frame and the patch side information for the patch in the reference frame is decoded. For example, a differential value is encoded for various patch coordinates such as an x-coordinate of the patches (e.g., u0), a y-coordinate of the patches (e.g., v0), an x-coordinate of the patches in 3D (e.g., u1), a y-coordinate of the patches in 3D (e.g., v1), and a z-coordinate of the patches in 3D (e.g., d1). In addition, a change in the x-coordinate of the current patch relative to the previous patch (e.g., deltaSizeU0) from the reference frame and a change in the y-coordinate of the current patch relative to the previous patch (e.g., deltaSizeV0) in the reference frame is decoded. In addition patch side information for the patch in the current frame is inherited from the patch side information for the patch in the reference frame for various patch coordinates such as a projection direction (e.g. patchNormalAxis), scale and rotation parameters (e.g. patchRotation, patchScale) and patch orientation (e.g patchorientationIndex).

[00172] Following the execution of block 2014, the method 2000 returns to block 2006 to continue the iterative process for the next patches in the patches buffer.

[00173] Referring back to block 2010, when a match does not exist (e.g., the value of the best match index equals -1), then the method 2000 proceeds to block 2016. In block 2016, Anchor's method is performed.

[00174] The following tables may be used for syntax elements.

Table 1 – GroupofFrames header – single frame

if (GOF > 1){	
numGlobalMatchedPatches	***ReadUint32()***
}	
if (numGlobalMatchedPatches > 0) {	
Iterate GOF frames : f	
if (f = = 0) {	
Occupancy maps x 1	*ReadOccupancyMap1()*
} else {	
Occupancy maps x (GOF-1)	*ReadOccupancyMap2()*
} else {	
Occupancy maps x GOF	*ReadOccupancyMap1()*
}	

Table 2 – GroupofFrames header – single frame

GroupOf Frames header	
if (GOF > 1) {	
numGlobalMatchedPatches	*ReadUint32()*
}	
if (numGlobalMatchedPatches > 0) {	
Iterate GOF frames : f	
Occupancy maps x 1	*ReadOccupancyMap1() for first frame*
Occupancy maps x (GOF-1)	*ReadOccupancyMap2() for left (GOF-1) frames*
} else {	
Iterate GOF frames : f	
Occupancy maps x 1	*ReadOccupancyMap1() for first frame*
Occupancy maps x (GOF-1)	*ReadOccupancyMap3() for left (GOF-1) frames*
}	

Table 3 – ReadOccuancyMap1()function

Patch count	ReadUint32
Occupancy precision	ReadUint8
Max candidate count	ReadUint8
bitCountU0	ReadUint8
bitCountV0	ReadUint8
bitCountU1	ReadUint8
bitCountV1	ReadUint8
bitCountD1	ReadUint8
Arithmetic bitstream size	ReadUint32
Arithmetic bitstream	*ReadPatchArithmetic1() for all patches*
Arithmetic bitstream	**ReadArithmetic()**

Table 4 – ReadOccuancyMap2()function

PatchCount	ReadUint32
occupancyPrecision	ReadUint8
maxCandidateCount	ReadUint8
bitCountU0	ReadUint8
bitCountV0	ReadUint8
bitCountU1	ReadUint8
bitCountV1	ReadUint8
bitCountD1	ReadUint8
Arithmetic bitstream size	ReadUint32
Arithmetic bitstream	***ReadPatchArithmetic2()*** *for the first numGlobalMatchedPatches patches*
Arithmetic bitstream	***ReadPatchArithmetic3()*** *for the left (patchCount - numGlobalMatchedPatches) patches*
Arithmetic bitstream	ReadArithmetic()

Table 5 – ReadArithmetic() function

patch.bestMatchedIndex + 1	*DecodeUInt32(bitCountPrePatchCount)*
if (patch.bestMatchedIndex == -1) {	
Arithmetic bitstream	*ReadPatchArithmetic1()*
} else {	
Arithmetic bitstream	*ReadPatchArithmetic2()*
}	
patch.bestMatchedIndex + 1	*DecodeUInt32(bitCountPrePatchCount)*
}	
// Block to patch index decoding	
For all patches	
if (!bestMatchedIndex) {	
U0	DecodeUInt32(bitCountU0)
V0	DecodeUInt32(bitCountV0)
U1	DecodeUInt32(bitCountU1)
V1	DecodeUInt32(bitCountV1)
D1	DecodeUInt32(bitCountD1)
deltaSizeU0	DecodeExpGolomb()
deltaSizeV0	DecodeExpGolomb()
} else {	
diff_U0	*DecodeUInt32(bitCountU0)*
diff_V0	*DecodeUInt32(bitCountV0)*
diff_U1	*DecodeUInt32(bitCountU1)*
diff_V1	*DecodeUInt32(bitCountV1)*
diff_D1	*DecodeUInt32(bitCountD1)*
diff_deltaSizeU0	*DecodeExpGolomb()*
diff_deltaSizeV0	*DecodeExpGolomb()*

}	
// Block to patch index decoding	
For all blocks	
If number of candidate patches > 1	
Candidate index	Decode
If Candidate index == maxCandidateCount	
block to patch index	DecodeUInt32(bitCountPatch)
Else	
Block to patch index = Candidate index	
// Occupancy map decoding	
For all blocks	
If Block to patch index > 0	
isFull	decode
If not Full	
bestTraversalOrderIndex	decode
runCountMinusTwo	decode
Occupancy	decode
for (size_t r = 0; r < runCountMinusOne; ++r)	
{	
runLength	decode
for (size_t j = 0; j <= runLength; ++j)	
Block[traversalOrder[i++]] = occupancy;	
occupancy = !occupancy;	
}	
For all remaining blocks	
Block[traversalOrder[i++]] = occupancy;	

[00175] From the foregoing, it should be appreciated that the disclosed embodiments provide significant improvements over the current technology. For example, the patch allocation process is based not only on geometric characteristics of the original point cloud, e.g., spatial position and normal deviation for adjacent points, but it also considers temporal correlation in a series of frames. The disclosed methods are easy to track by patch position analysis in a geometry and/or texture map. In addition, auxiliary information coding permits patch side information to be encoded using a difference between values based on a patch index value for different temporal images.

[00176] While several embodiments have been provided in the present disclosure, it may 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 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.

[00177] In addition, techniques, systems, subsystems, and methods described and illustrated in the various embodiments as discrete or separate may be combined or integrated with other systems, components, techniques, or methods without departing from the scope of the present disclosure. Other examples of changes, substitutions, and alterations are ascertainable by one skilled in the art and may be made without departing from the spirit and scope disclosed herein.

CLAIMS

What is claimed is:

1. A method of media coding implemented by a coding device, comprising:

generating, using the coding device, a patch list identifying matched patches between a current patch frame and a reference patch frame; and

encoding, using the coding device, a difference between patch side information of a first patch in the current patch frame and patch side information of a second patch in the reference patch frame when a best match patch index value for the first patch in the patch list of the current patch frame matches a second patch index value in a second patch list of the reference patch frame.

2. A method of media coding implemented by a coding device, comprising:

obtaining patch side information for all patches in a current frame;

encoding patch parameters corresponding to the patches;

iterating a patches buffer containing a list of the patches, the patches buffer organized based on matches between consecutive frames;

encoding a value in a best matches index of the patches buffer for a first patch;

determining that there is a match between the current frame and a reference frame using the best matches index;

obtaining the patch side information from a reference frame; and

encoding a difference between the patch side information for the first patch in the current frame and the patch side information from the reference frame.

3. A method of media coding implemented by a coding device, comprising:

obtaining patch side information for all patches in a current frame;

encoding patch parameters corresponding to the patches;

iterating a patches buffer containing a list of the patches, the patches buffer organized based on matches between consecutive frames;

determining that the patches buffer includes a global matched patch;

determining that there is a match between the current frame and a reference frame using the best matches index when the patches buffer includes the global matched patch;

obtaining the patch side information from a reference frame; and

encoding a difference between the patch side information for the first patch in the current frame and the patch side information from the reference frame when the match has been determined.

4. A method of media coding implemented by a coding device, comprising:

obtaining patch side information for all patches in a group of frames;

encoding global matched patches for the group of frames;

determining whether there are global matched patches for the group of frames;

iterate each frame in the group of frames and a patches buffer for the current frame;

encode a difference between patch side information for a first patch in the current frame and the patch side information from a reference frame when there are global matched patches;

iterate each frame in the group of frames and a patches buffer for the current frame; and

encode:

a difference between the patch side information for the patch in the current frame and the patch side information for the patch in the reference frame when there is a local matched patch; or

patch coordinates for the patch in the current frame, a difference between a size of the patch and a size of a previously encoded patch within the current frame; and a normal axis for a current patch.

5. A method of media coding implemented by a coding device, comprising:

obtaining patch side information for all patches in a current frame from an encoded point cloud stream;

obtaining patch side information for all patches in a current frame from an encoded point cloud stream;

decoding patch parameters corresponding to the patches; iterating a patches buffer containing a list of the patches, the patches buffer organized based on matches between consecutive frames;

decoding a value in a best matches index of the patches buffer for a first patch;

determining that there is a match between the first patch in the current frame and a second patch in the reference frame using the best matches index; obtaining the patch side information from a reference frame; and

decoding a difference between the patch side information for the first patch in the current frame and the patch side information for the second patch from the reference frame.

6. The method of any of claims 1 to 5, wherein the coding device is a video encoder or a video decoder.

7. The method of any of claims 1 to 6, wherein the patch side information includes one or more of a height, a width, and a depth of the patches.

8. The method of any of claims 1 to 7, wherein the patch parameters include one or more of a patch count, an occupancy precision, a maximum candidate count, a bit count for an x-coordinate in two dimensions (2D), a bit count for a y-coordinate in 2D, a bit count for an x-coordinate in three dimensions (3D), a bit count for a y-coordinate in 3D, and a bit count for a z-coordinate in 3D.

9. The method of claim 8, wherein the patch count represents a number of patches for an image.

10. The method of claim 8, wherein the occupancy count represents an amount of downsampling used.

11. The method of claim 8, wherein the maximum candidate count represents a number of patches that belong to a same block of the occupancy precision.

12. The method of claim 8, wherein the bit count represents a number of bits needed to describe a coordinate.

13. The method of any of claims 1 to 12, wherein the patches buffer is organized based on any matches between consecutive frames.

14. The method of any of claims 1 to 13, wherein the global matched patches are stored at a beginning of the patches buffer.

15. The method of any of claims 1 to 14, wherein the local matched patches are stored after the global matched patches in the patches buffer.

16. The method of any of claims 1 to 15, wherein unmatched patches are stored after the global matched patches and the local matched patches in the patches buffer.

17. The method of any of claims 1 to 16, wherein one or more parameters from the patch side information of the reference frame are inherited for the patch side information of the current frame.

18. The method of any of claims 1 to 17, wherein one or more metadata parts from the reference frame are inherited for the current frame.

19. The method of any of claims 1 to 18, wherein the one or more metadata parts include a patch rotation, a scale parameter, and a material identifier.

20. A coding apparatus, comprising:

a receiver configured to receive a picture to encode or to receive a bitstream to decode;

a transmitter coupled to the receiver, the transmitter configured to transmit the bitstream to a decoder or to transmit a decoded image to a display;

a memory coupled to at least one of the receiver or the transmitter, the memory configured to store instructions; and

a processor coupled to the memory, the processor configured to execute the instructions stored in the memory to perform the method in any of claims 1 to 19.

21. The coding apparatus of claim 20, further comprising a display configured to display an image.

22. A system, comprising:

an encoder; and

a decoder in communication with the encoder, wherein the encoder or the decoder includes the coding apparatus of any of claims 19 to 20.

23. A means for coding, comprising:

receiving means configured to receive a picture to encode or to receive a bitstream to decode;

transmission means coupled to the receiving means, the transmission means configured to transmit the bitstream to a decoder or to transmit a decoded image to a display means;

storage means coupled to at least one of the receiving means or the transmission means, the storage means configured to store instructions; and

processing means coupled to the storage means, the processing means configured to execute the instructions stored in the storage means to perform the method in any of claims 1 to 19.

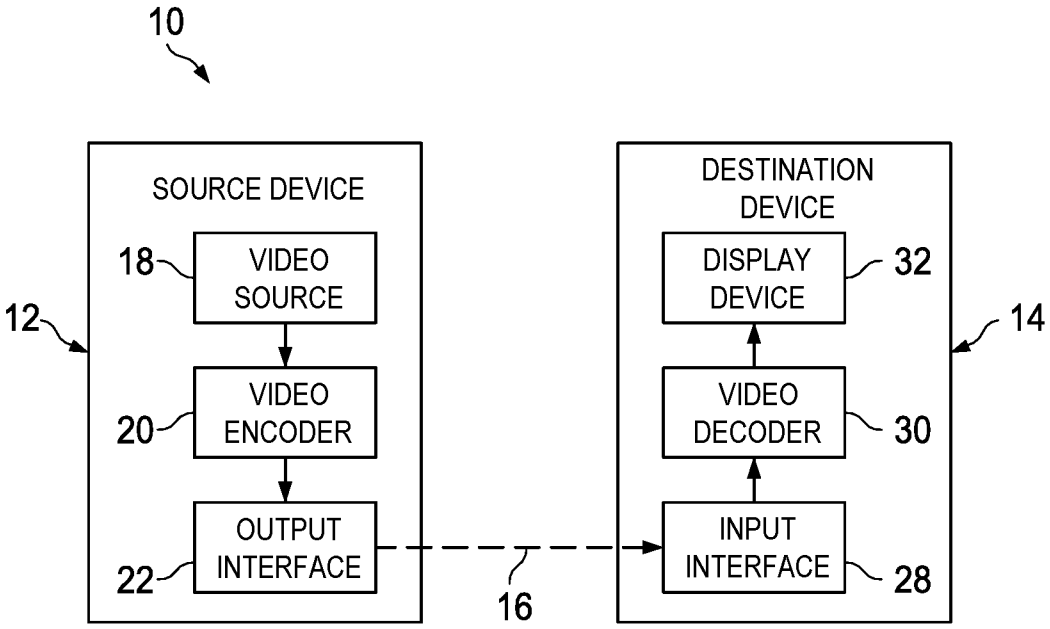

10
SOURCE DEVICE
18
VIDEO SOURCE
12
20
VIDEO ENCODER
22
OUTPUT INTERFACE
16
DESTINATION DEVICE
DISPLAY DEVICE
32
14
VIDEO DECODER
30
INPUT INTERFACE
28

FIG. 1

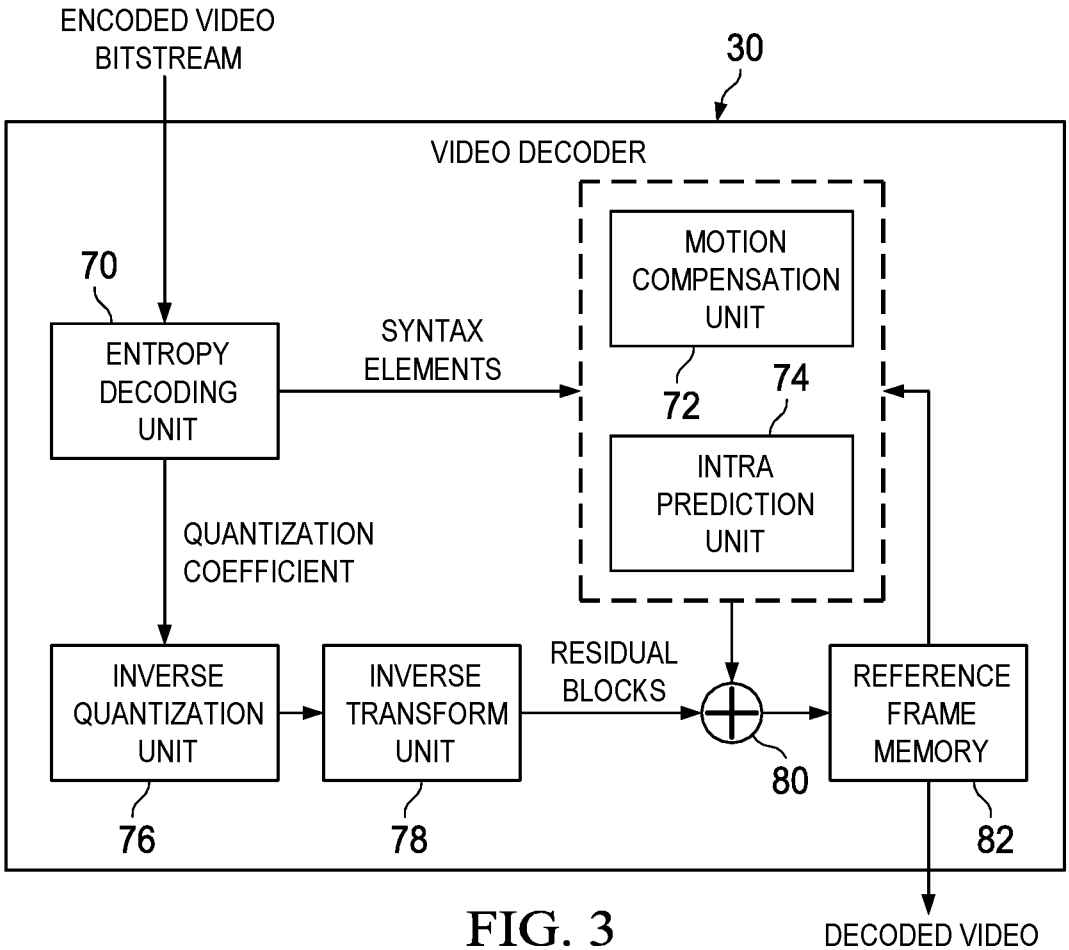

ENCODED VIDEO BITSTREAM
30
VIDEO DECODER
70
ENTROPY DECODING UNIT
SYNTAX ELEMENTS
MOTION COMPENSATION UNIT
72
74
INTRA PREDICTION UNIT
QUANTIZATION COEFFICIENT
INVERSE QUANTIZATION UNIT
INVERSE TRANSFORM UNIT
RESIDUAL BLOCKS
80
REFERENCE FRAME MEMORY
76
78
82
DECODED VIDEO

FIG. 3

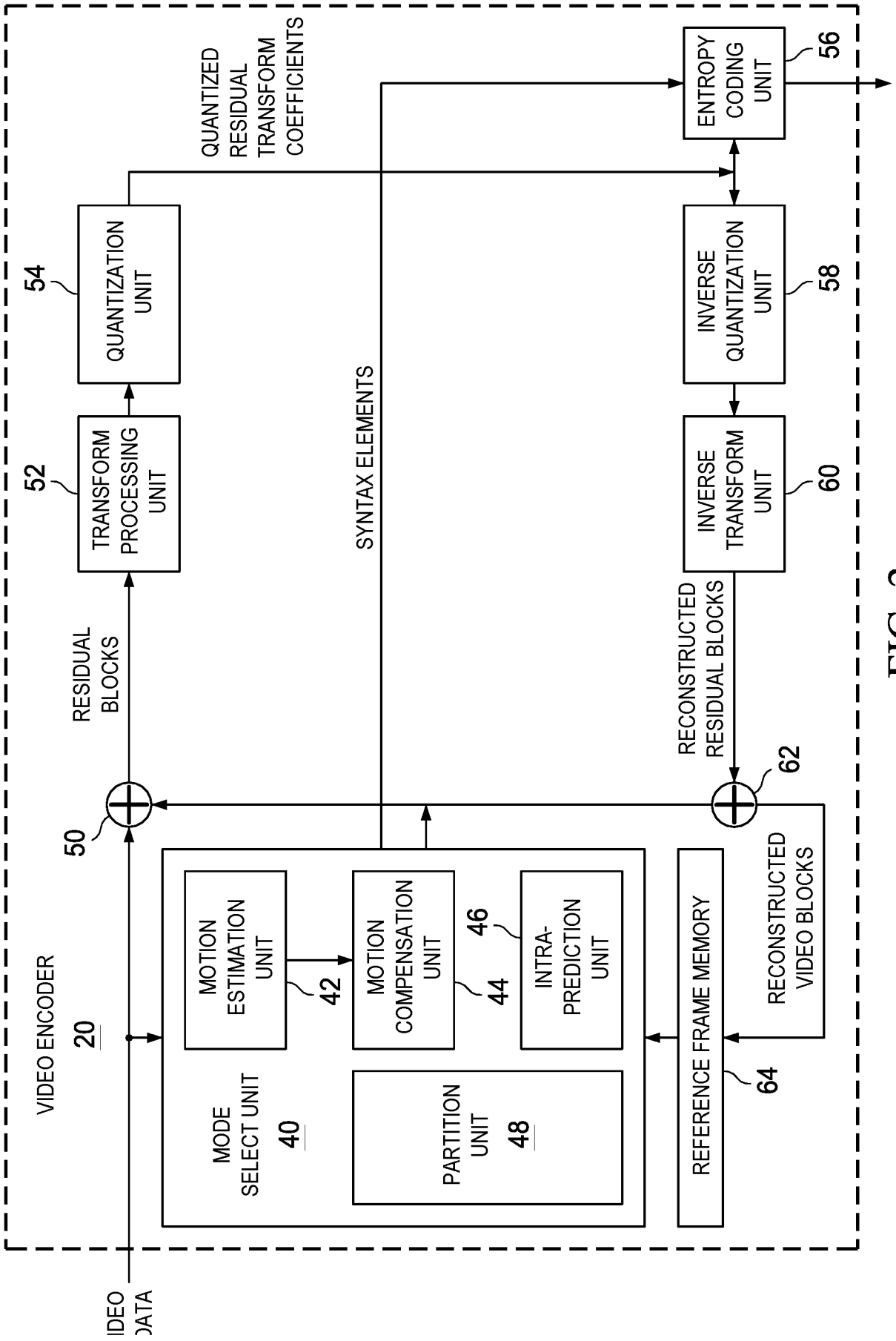

VIDEO ENCODER
20
VIDEO DATA
MODE SELECT UNIT
40
MOTION ESTIMATION UNIT
42
MOTION COMPENSATION UNIT
44
INTRA-PREDICTION UNIT
46
PARTITION UNIT
48
50
RESIDUAL BLOCKS
TRANSFORM PROCESSING UNIT
52
QUANTIZATION UNIT
54
QUANTIZED RESIDUAL TRANSFORM COEFFICIENTS
SYNTAX ELEMENTS
ENTROPY CODING UNIT
56
INVERSE QUANTIZATION UNIT
58
INVERSE TRANSFORM UNIT
60
RECONSTRUCTED RESIDUAL BLOCKS
62
RECONSTRUCTED VIDEO BLOCKS
REFERENCE FRAME MEMORY
64

FIG. 2

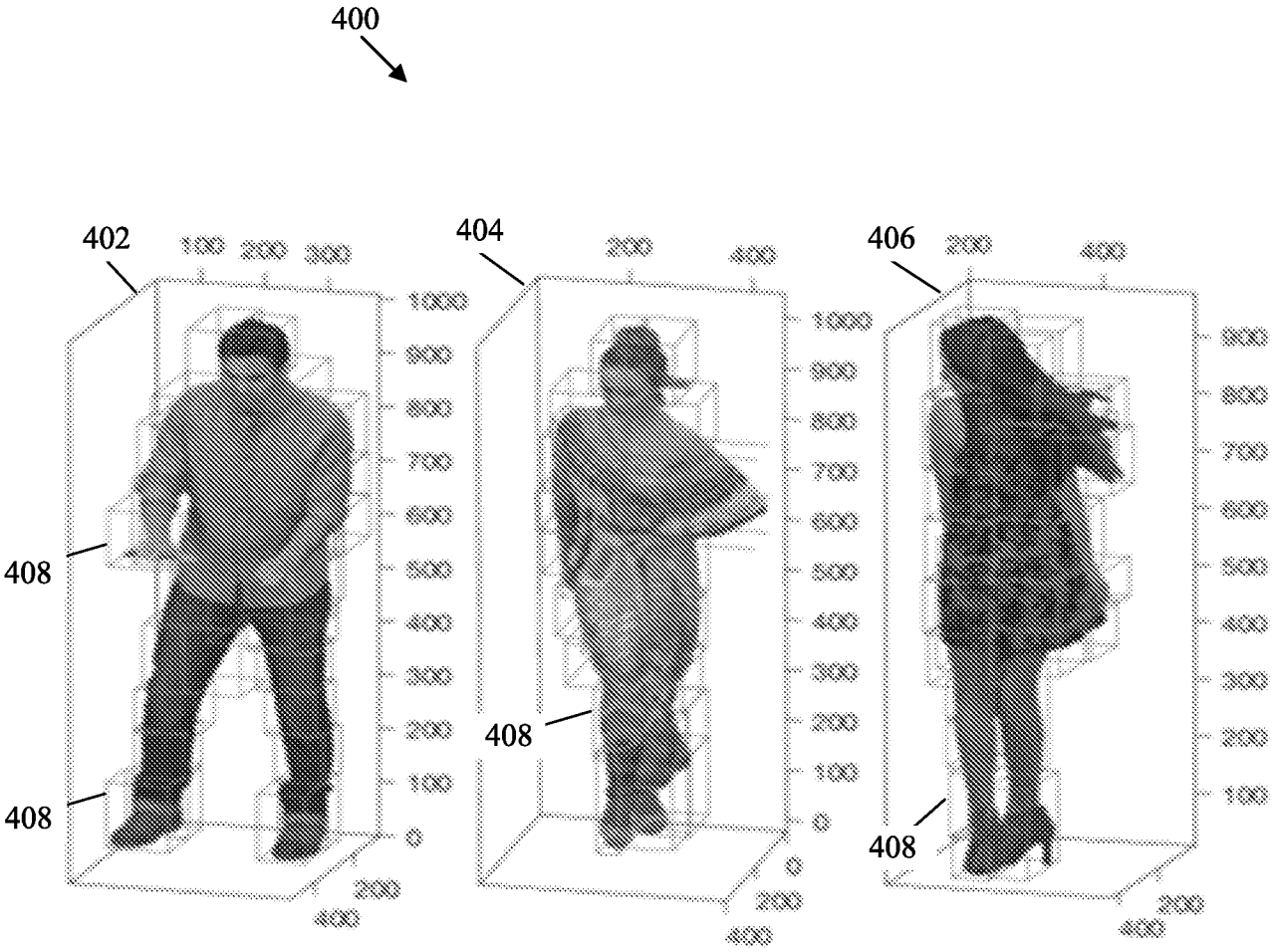

400
402
404
406
408
408
408
408
100
200
300
200
400
200
400
1000
900
800
700
600
500
400
300
200
100
0
200
400

FIG. 4

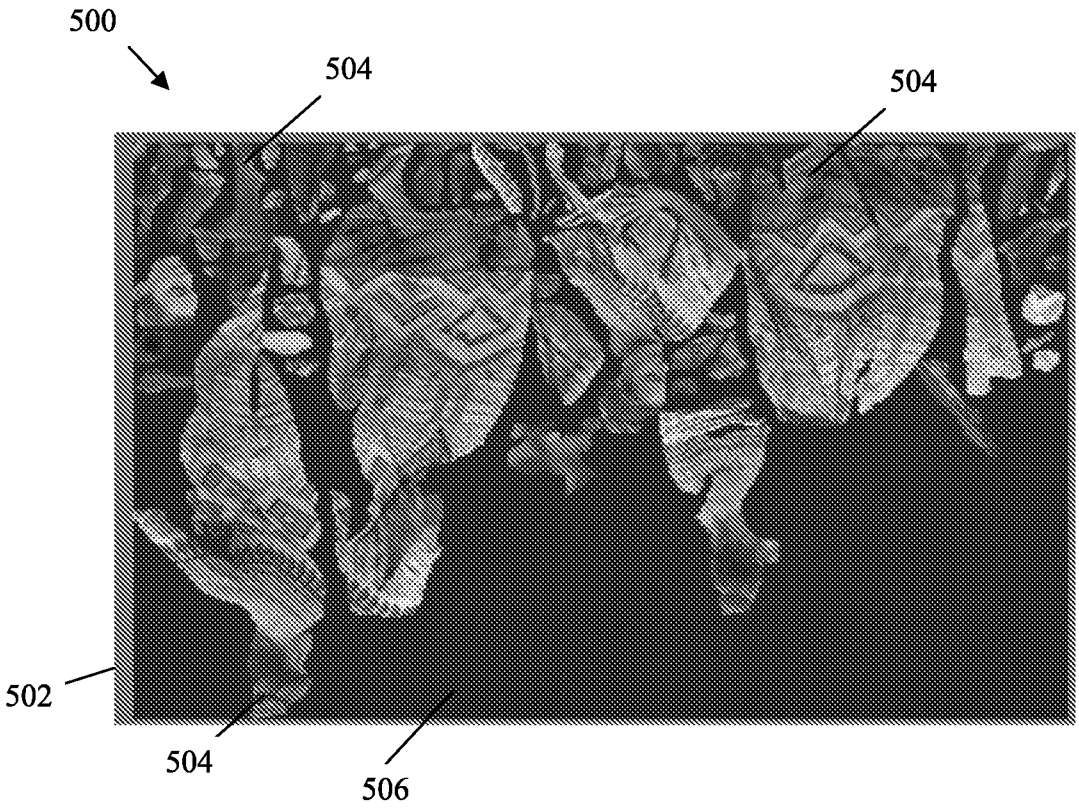


FIG. 5

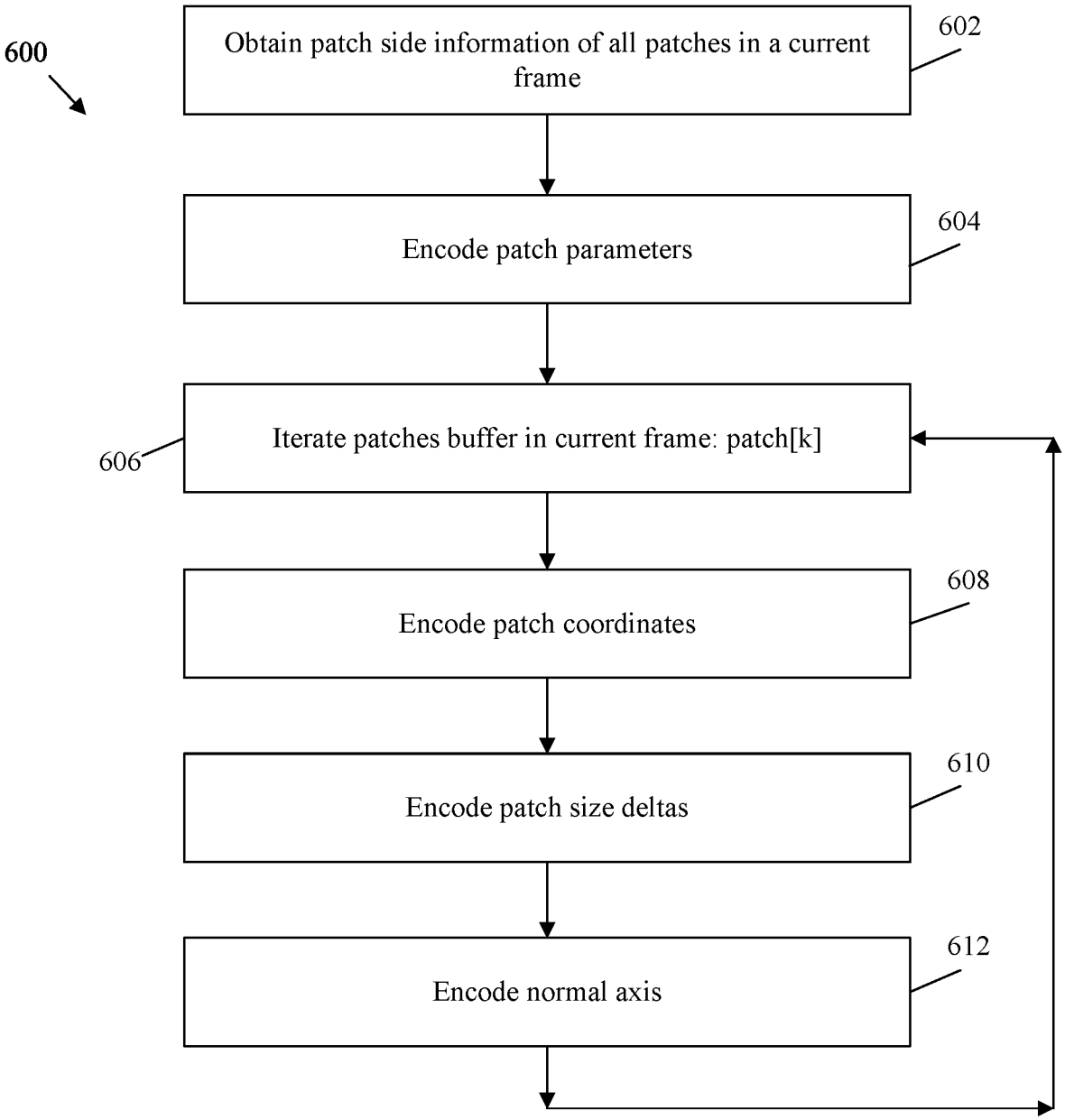
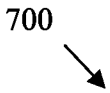

600
Obtain patch side information of all patches in a current frame
602
Encode patch parameters
604
Iterate patches buffer in current frame: patch[k]
606
Encode patch coordinates
608
Encode patch size deltas
610
Encode normal axis
612

FIG. 6



Algorithm 1. Patch side information coding
1. Encode patchCount, occupancyPrecision and maxCandidateCount with number of bits.
2. Encode bitCountU0, bitCountV0, bitCountU1, bitCountV1 and bitCountD1 with number of bits.
3. **while** patch[k] in patches buffer **do**
Encode u0,v0,u1,v1,d1 of patch[k] with number of bits
deltaSizeU0=sizeU0 – preSizeU0;
deltaSizeV0=sizeV0 – preSizeV0;
Encode deltaSizeU0 and deltaSizeV0 with arithmetic codec or exp golomb codec.
Encode normalAxis of the patch with number of bits.
Update preSizeU0 and preSizeV0.
end **while**

FIG. 7

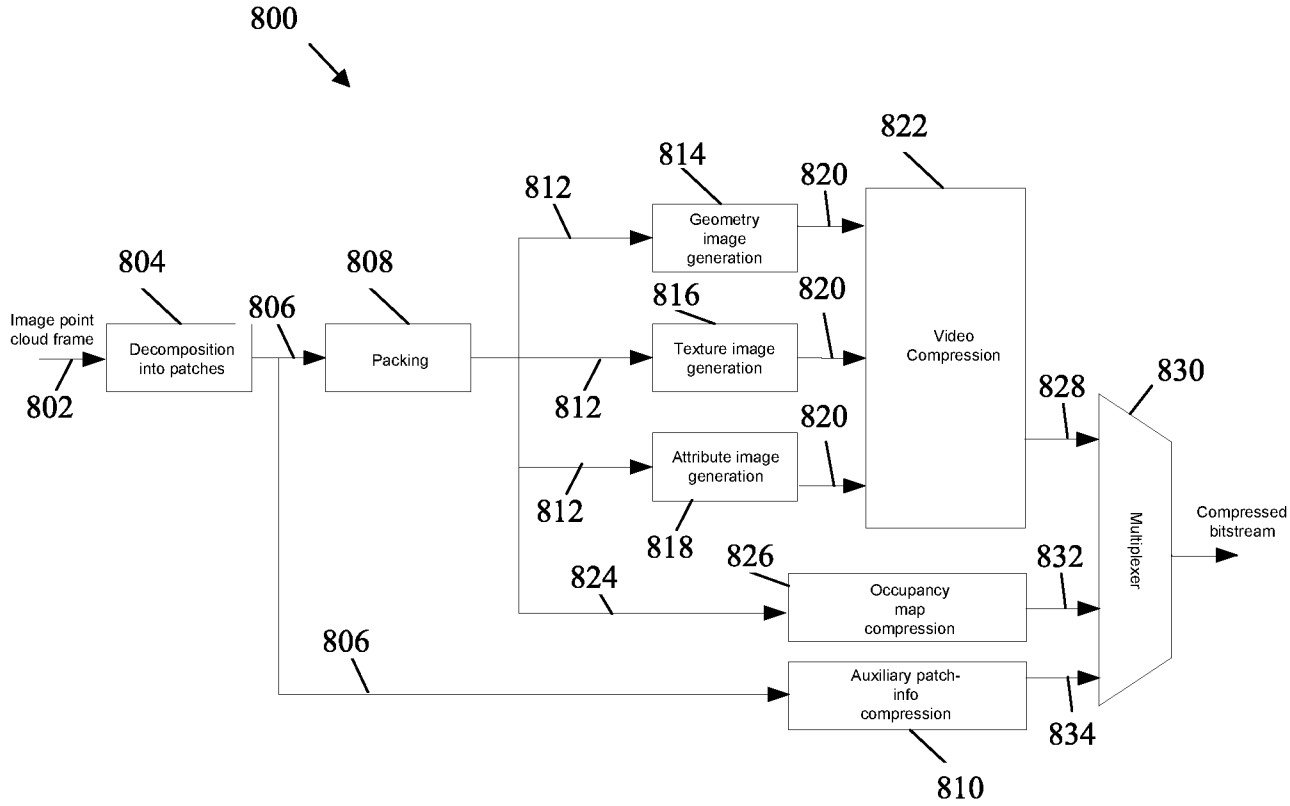

800
Image point cloud frame
802
Decomposition into patches
804
806
Packing
808
812
Geometry image generation
814
Texture image generation
816
Attribute image generation
818
820
Video Compression
822
Occupancy map compression
824
826
Auxiliary patch-info compression
810
828
Multiplexer
830
832
834
Compressed bitstream

FIG. 8

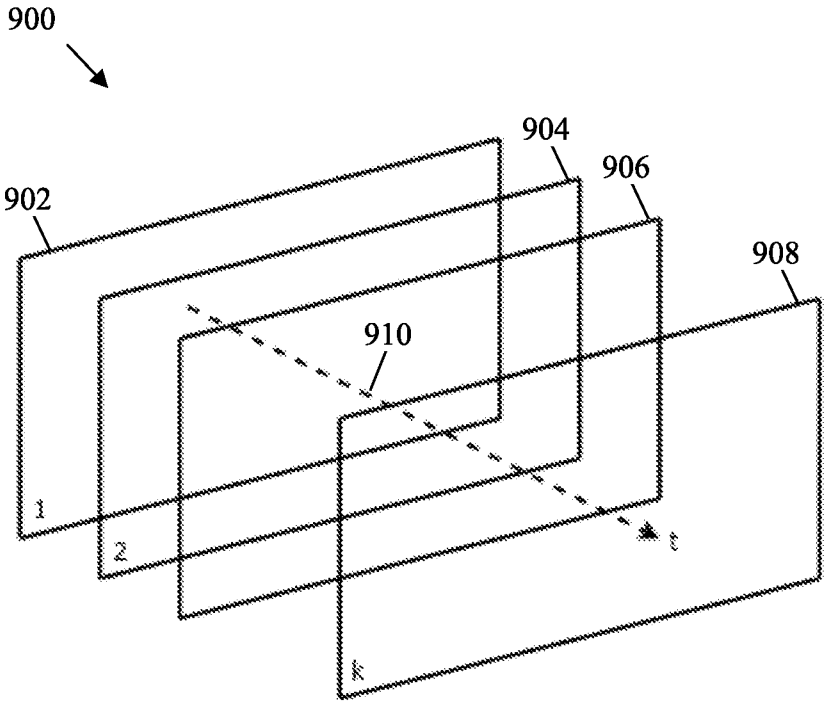

900
902
904
906
908
910
1
2
k
t

FIG. 9

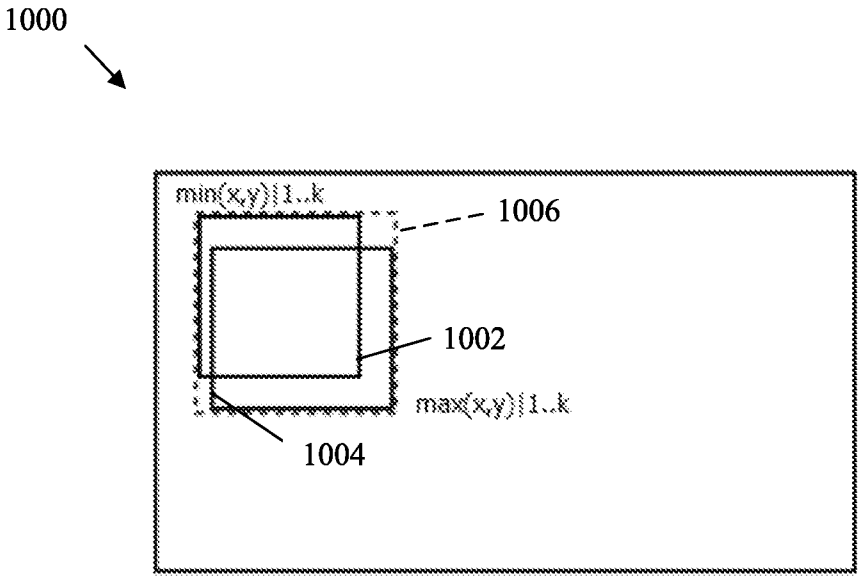

1000
min(x,y)|1..k
1006
1002
max(x,y)|1..k
1004

FIG. 10

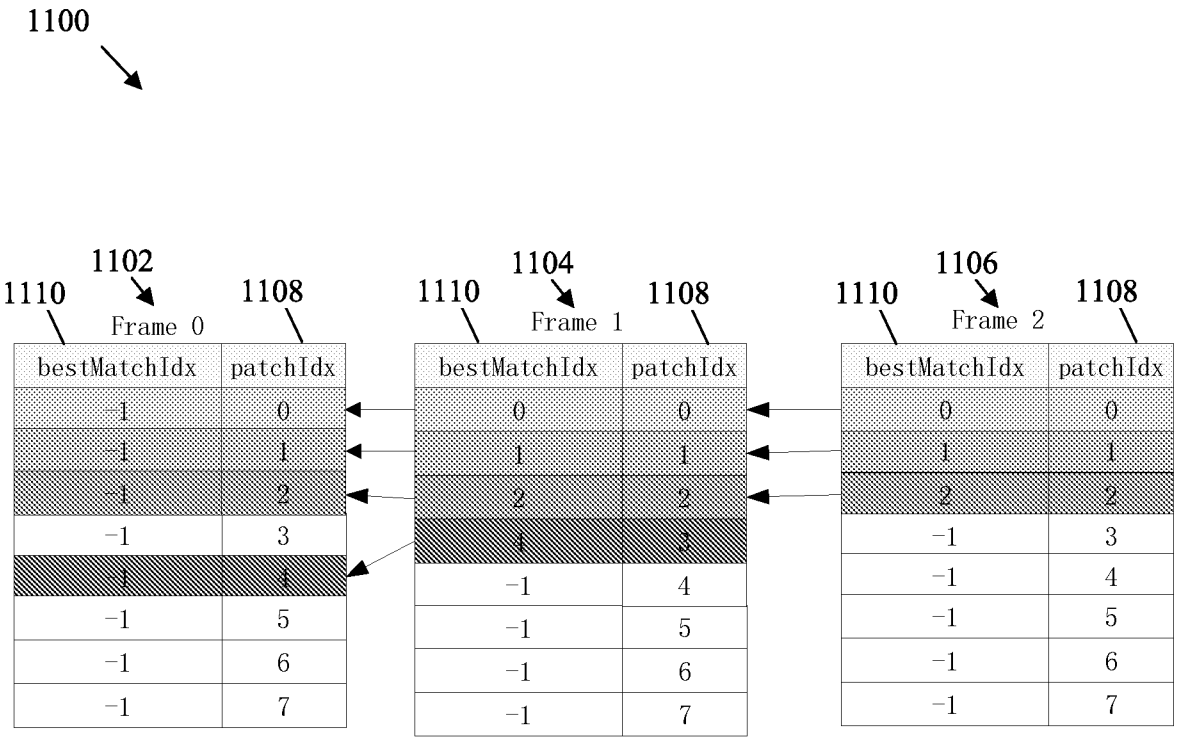

1100
1102
1104
1106
1110
1108
Frame 0
Frame 1
Frame 2
bestMatchIdx
patchIdx
-1 0
-1 1
-1 2
-1 3
-1 4
-1 5
-1 6
-1 7
0 0
1 1
2 2
4 3
-1 4
-1 5
-1 6
-1 7
0 0
1 1
2 2
-1 3
-1 4
-1 5
-1 6
-1 7

FIG. 11

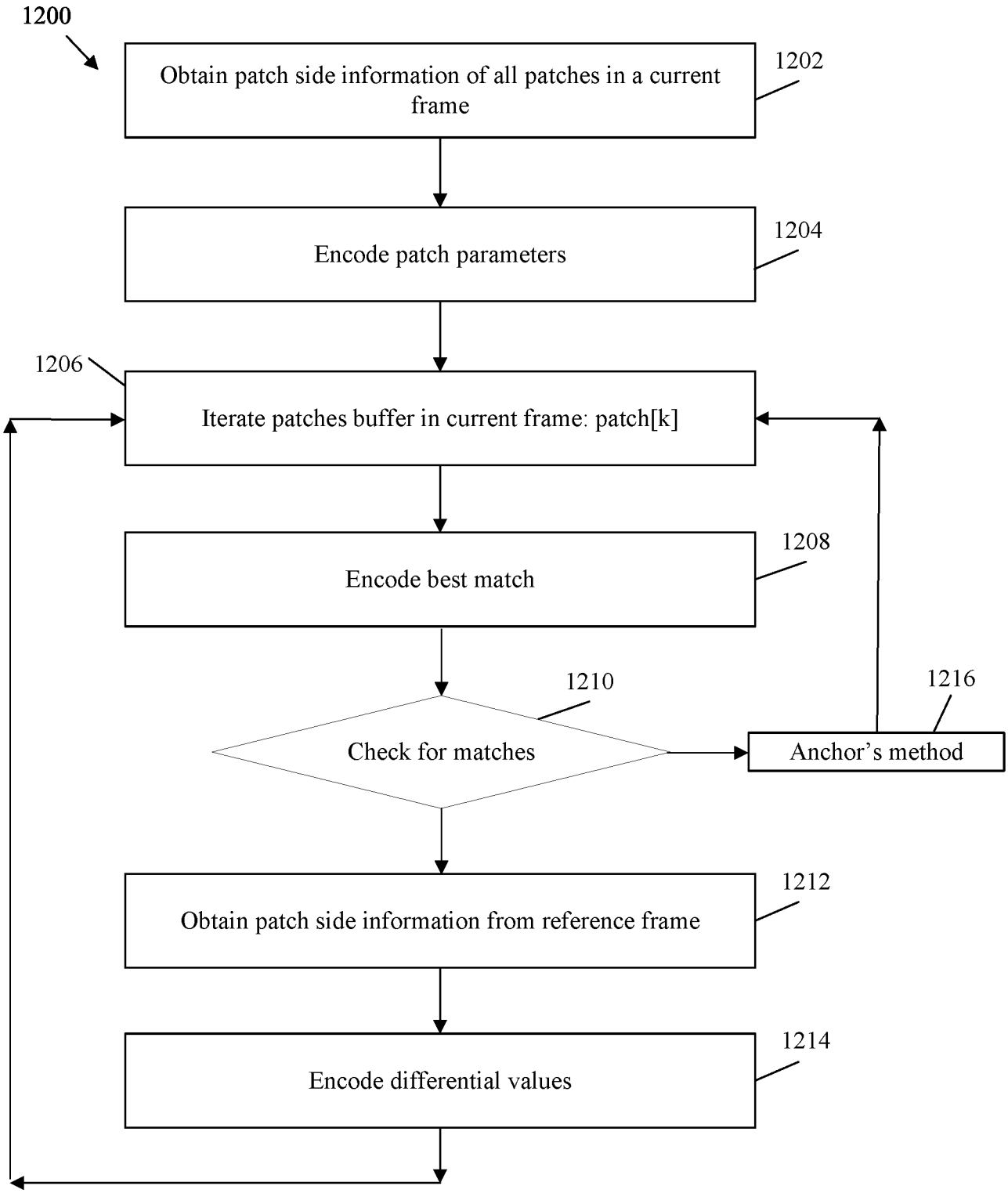

1200
Obtain patch side information of all patches in a current frame
1202
Encode patch parameters
1204
1206
Iterate patches buffer in current frame: patch[k]
Encode best match
1208
1210
Check for matches
1216
Anchor's method
Obtain patch side information from reference frame
1212
Encode differential values
1214

FIG. 12

1300

Algorithm 2. Patch side information coding method for a pair of frames
1. Encode patchCount, occupancyPrecision and maxCandidateCount with number of bits.
2. Encode bitCountU0, bitCountV0, bitCountU1, bitCountV1, bitCountD1 with number of bits.
3. while patch[k] in patches buffer do
Encode (patch[k].bestMatchedIndex + 1) with number of bits.
if patch[k]. bestMatchedIndex == -1,
Call anchor's method Algorithm1.
else
Get prePatch in previous frame with patch[k]. bestMatchedIndex.
Compute differential value for
u0,v0,u1,v1,d1,sizeU0, sizeV0
between patch[k] and prePatch.
diff_u0= patch[k].u0 – prePatch.u0,
diff_v0= patch[k].v0 – prePatch.v0,
diff_u1= patch[k].u1 – prePatch.u1,
diff_v1= patch[k].v1 – prePatch.v1,
diff_d1= patch[k].d1 – prePatch.d1,
diff_sizeU0= patch[k]. sizeU0– prePatch. sizeU0,
diff_sizeV0= patch[k]. sizeV0– prePatch. sizeV0,
Encode
diff_u0, diff_v0, diff_u1, diff_v1, diff_d1, diff_sizeU0, diff_sizeV0
with encoding method.
end if
end while

FIG. 13

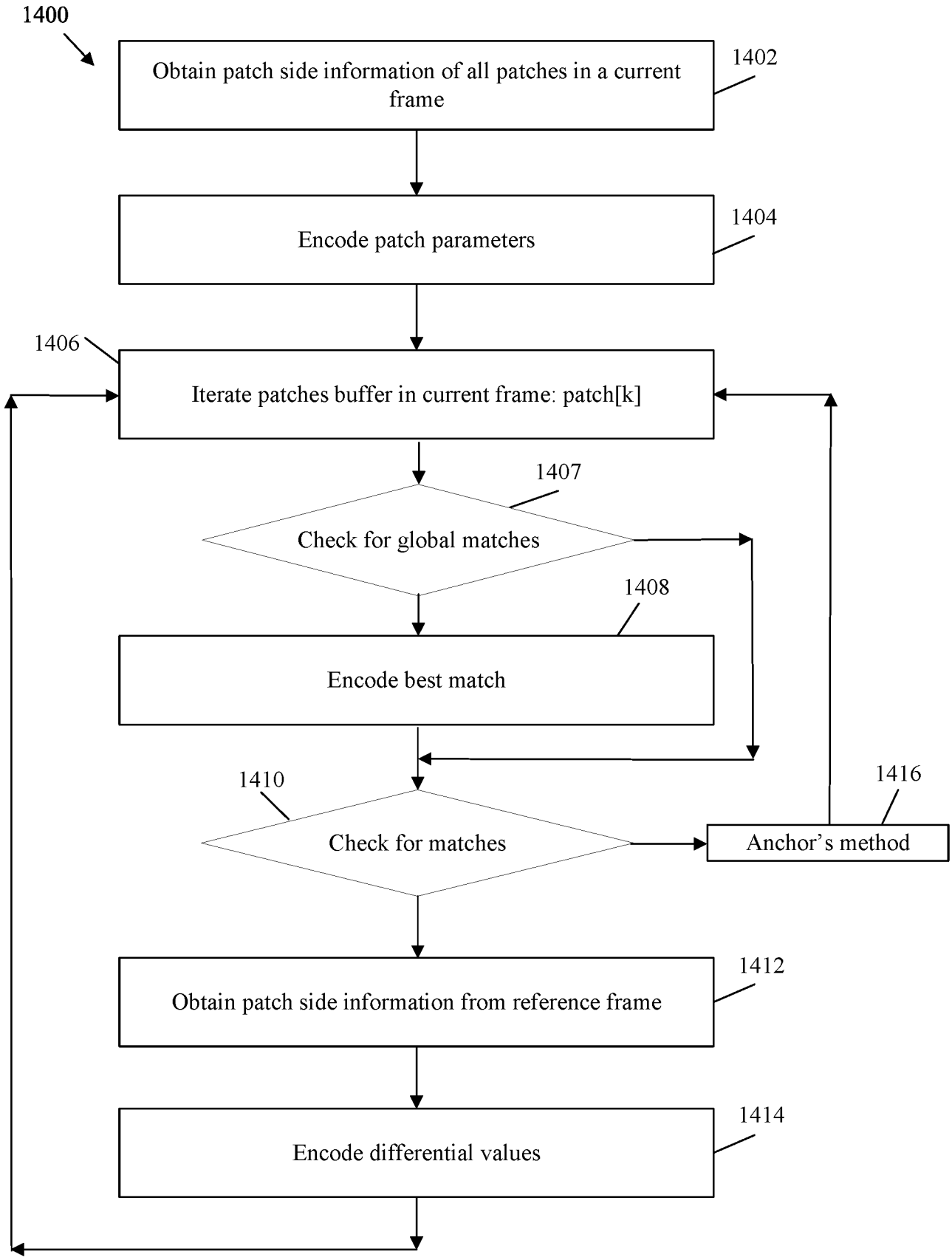

1400
Obtain patch side information of all patches in a current frame
1402
Encode patch parameters
1404
1406
Iterate patches buffer in current frame: patch[k]
1407
Check for global matches
1408
Encode best match
1410
Check for matches
1416
Anchor's method
1412
Obtain patch side information from reference frame
1414
Encode differential values

FIG. 14

1500

```
Algorithm 3. Patch side information coding method for multiple frames
1.        Encode patchCount, occupancyPrecision, maxCandidateCount with
number of bits.
2.        Encode bitCountU0, bitCountV0, bitCountU1, bitCountV1 bitCountD1
with number of bits.
3.        while patch[k] in patches buffer do
    if k >= numGlobalMatchedPatches,
        Encode (patch[k].bestMatchedIndex + 1) with number of bits.
    end if
    if patch[k]. bestMatchedIndex == -1,
    Use orinal patch compression method.
 else
   Get prePatch in previous frame with patch[k]. bestMatchedIndex.
Compute differential value of u0,v0,u1,v1,d1,sizeU0, sizeV0 between patch[k]
and prePatch.
diff_u0= patch[k].u0 – prePatch.u0,
diff_v0= patch[k].v0 – prePatch.v0,
diff_u1= patch[k].u1 – prePatch.u1,
diff_v1= patch[k].v1 – prePatch.v1,
diff_d1= patch[k].d1 – prePatch.d1,
diff_sizeU0= patch[k]. sizeU0– prePatch. sizeU0,
diff_sizeV0= patch[k]. sizeV0– prePatch. sizeV0,
Encode
diff_u0, diff_v0, diff_u1, diff_v1, diff_d1, diff_sizeU0, diff_sizeV0
with encoding method.
 end if
  end while
```

FIG. 15

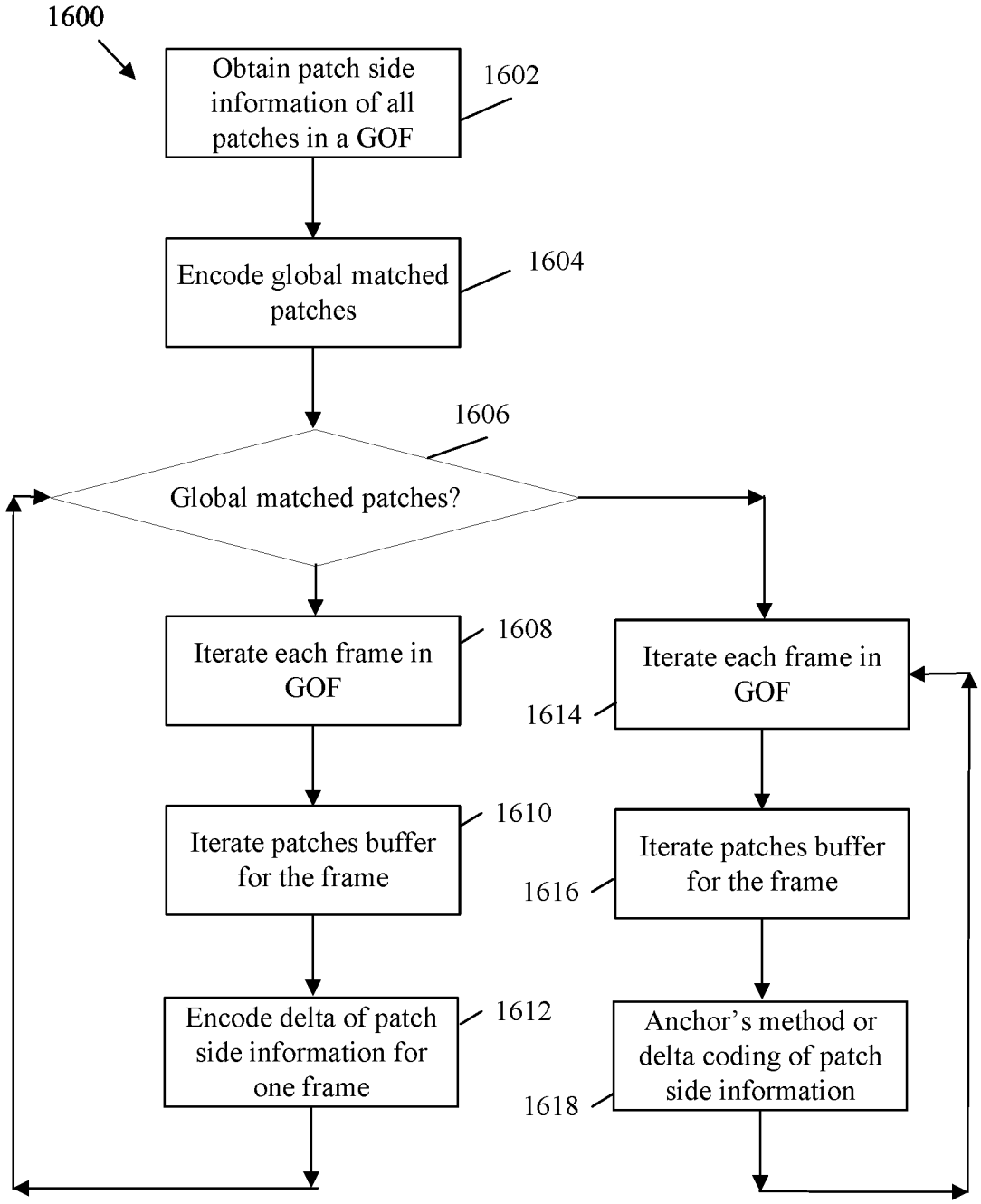

1600
Obtain patch side information of all patches in a GOF
1602
Encode global matched patches
1604
1606
Global matched patches?
Iterate each frame in GOF
1608
Iterate patches buffer for the frame
1610
Encode delta of patch side information for one frame
1612
Iterate each frame in GOF
1614
Iterate patches buffer for the frame
1616
Anchor's method or delta coding of patch side information
1618

FIG. 16

1700

Algorithm 4. Patch side information coding method for multiple frames

```
1.        Encode numGlobalMatchedPatches with
number of bits.
2.        if numGlobalMatchedPatches > 0 do
     while frame i in GOF frames do
    while patch[k] in patches buffer of frame i do
         Call Algorithm3
       end while
   end while
else
   while frame i in GOF frames do
     while patch[k] in patches buffer of frame i do
          Call Algorithm1 or Algorithm2
       end while
    end while
end if
```

FIG. 17

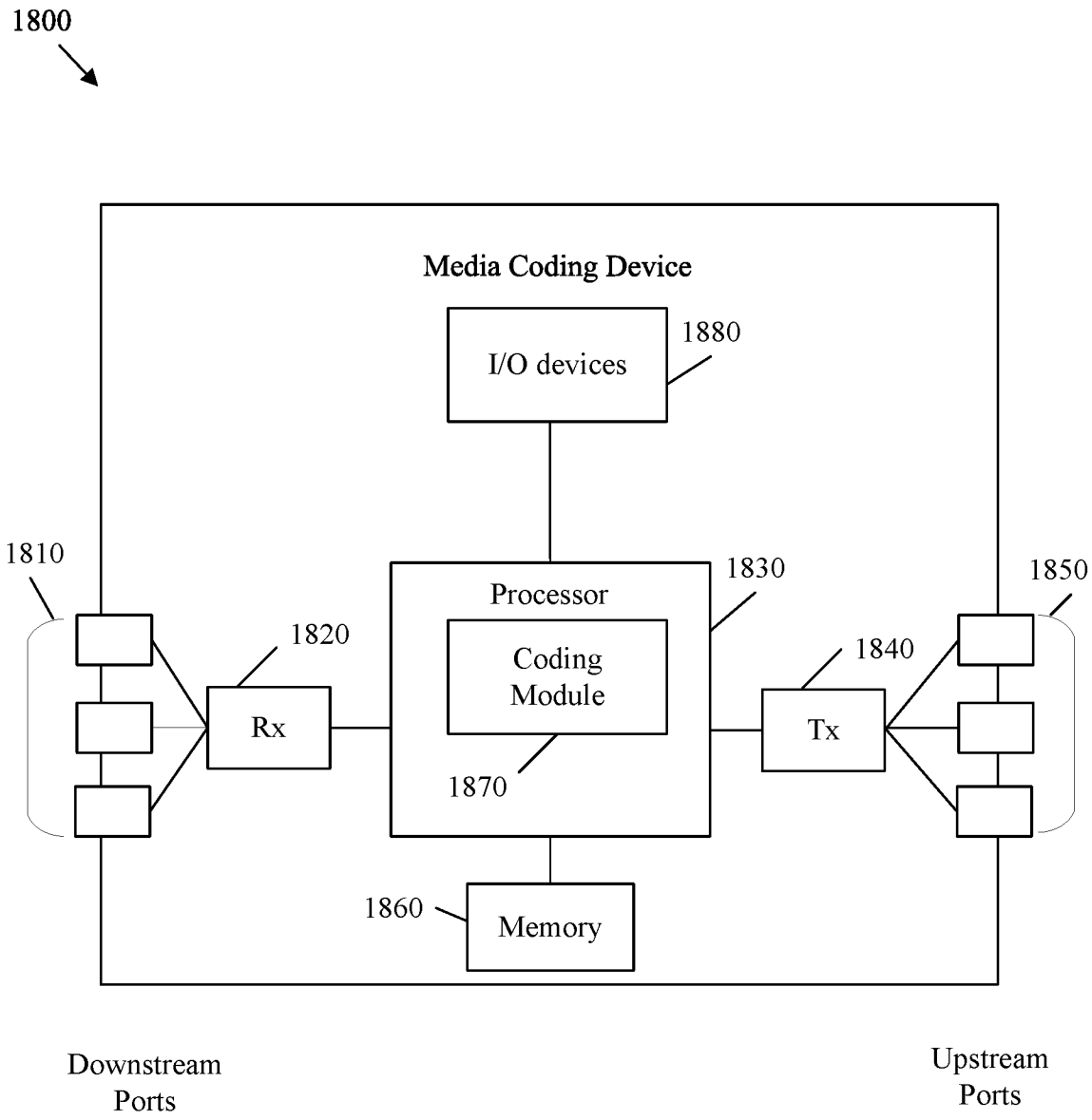

1800
Media Coding Device
I/O devices
1880
1810
1820
Rx
Processor
Coding Module
1870
1830
1840
Tx
1850
1860
Memory
Downstream Ports
Upstream Ports

FIG. 18

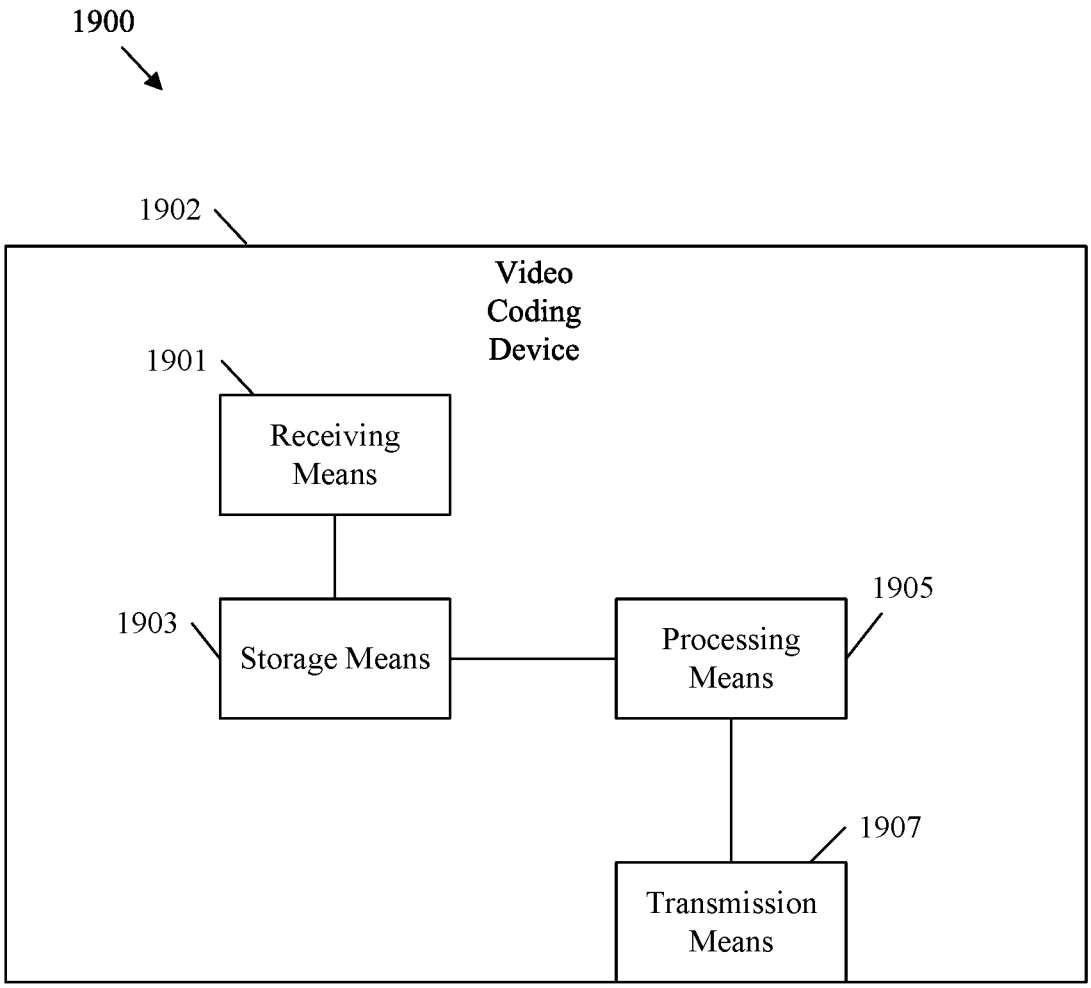

1900
1902
Video Coding Device
1901
Receiving Means
1903
Storage Means
1905
Processing Means
1907
Transmission Means

FIG. 19

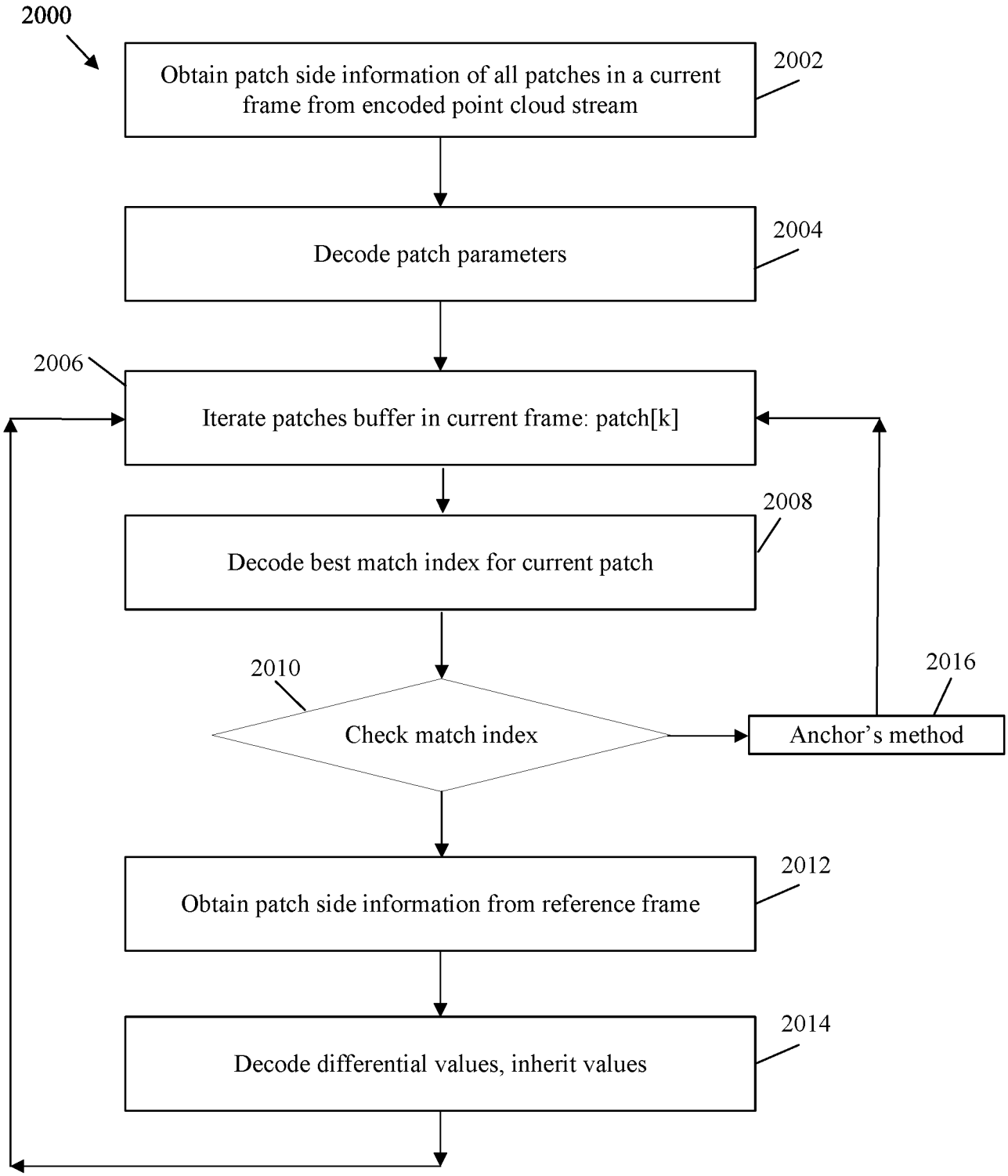

2000
Obtain patch side information of all patches in a current frame from encoded point cloud stream
2002
Decode patch parameters
2004
2006
Iterate patches buffer in current frame: patch[k]
2008
Decode best match index for current patch
2010
Check match index
2016
Anchor's method
2012
Obtain patch side information from reference frame
2014
Decode differential values, inherit values

FIG. 20

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/022776

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/597 H04N19/503
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	ZHANG DEJUN ET AL: "[PCC] TMC2 A differential coding method for patch side information", 122. MPEG MEETING; 16-4-2018 - 20-4-2018; SAN DIEGO; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m42761, 14 April 2018 (2018-04-14), XP030071100, the whole document -----	1-23
X	ZHANG DEJUN ET AL: "[PCC] TMC2 CE2.6 results", 122. MPEG MEETING; 16-4-2018 - 20-4-2018; SAN DIEGO; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m42712, 11 April 2018 (2018-04-11), XP030071051, the whole document -----	1-7, 20-23
	-/--	

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 June 2019	26/06/2019

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

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/022776

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ZHANG DEJUN ET AL: "[PCC] TMC2 Improved Temporally consistent Patch Packing", 122. MPEG MEETING; 16-4-2018 - 20-4-2018; SAN DIEGO; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m42718, 12 April 2018 (2018-04-12), XP030071057, the whole document	1-7, 13-16

X	DEJUN ZHANG: "A new patch side information encoding method for PCC TMC2", 121. MPEG MEETING; 22-1-2018 - 26-1-2018; GWANGJU; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m42195, 19 January 2018 (2018-01-19), XP030070537,	1-7,17, 18,20-23
Y	the whole document	8-12,19

A	"PCC Test Model Category 2 v0", 120. MPEG MEETING;23-10-2017 - 27-10-2017; MACAU; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. N17248, 15 December 2017 (2017-12-15), XP030023909, the whole document	1-23

Y	JULIEN RICARD ET AL: "Block to patch index coding", 122. MPEG MEETING; 16-4-2018 - 20-4-2018; SAN DIEGO; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m42629, 12 April 2018 (2018-04-12), XP030070968, the whole document	8-12

Y	MADHUKAR BUDAGAVI ET AL: "PCC Metadata", 122. MPEG MEETING; 16-4-2018 - 20-4-2018; SAN DIEGO; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m42725, 12 April 2018 (2018-04-12), XP030071064, the whole document	19
