



(51) International Patent Classification:

G06T 9/00 (2006.01) H04N 19/10 (2014.01)
H04N 19/136 (2014.01) H04N 19/46 (2014.01)
H04N 19/70 (2014.01)

(21) International Application Number:

PCT/US2023/032810

(22) International Filing Date:

15 September 2023 (15.09.2023)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

63/376,068 16 September 2022 (16.09.2022) US

(71) Applicant: **INNOPEAK TECHNOLOGY, INC.**
[US/US]; 2479 E. Bayshore RD., Ste. 110, Palo Alto, California 94303 (US).

(72) Inventors: **YU, Yue**; 2479 E. Bayshore RD., Ste. 110, Palo Alto, California 94303 (US). **YU, Haoping**; 2479 E. Bayshore RD., Ste. 110, Palo Alto, California 94303 (US).

(74) Agent: **LEE, Belinda**; 20 Pacifica, Suite 220, Irvine, California 92618 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

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

Declarations under Rule 4.17:

— of inventorship (Rule 4.17(iv))

Published:

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

(54) Title: POINT CLOUD DECODING METHOD, POINT CLOUD ENCODING METHOD, DECODER AND ENCODER

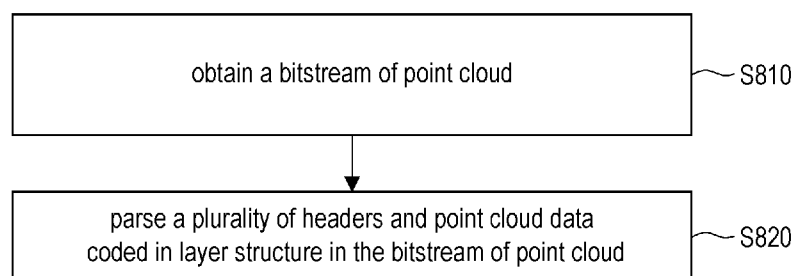


FIG. 8

(57) Abstract: A point cloud decoding method, a point cloud encoding method, a decoder and an encoder are provided. The point cloud decoding method includes the following step: obtaining a bitstream of point cloud; parsing a plurality of headers coded in layer structure in the bitstream of point cloud; and decoding the bitstream of point cloud according to the headers. The headers in the bitstream of point cloud comprises a sequence header, a geometry header, an attribute header, and a frame header. The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.



POINT CLOUD DECODING METHOD, POINT CLOUD ENCODING METHOD,
DECODER AND ENCODER

5 CROSS-REFERENCE TO RELATED APPLICATION

This application claims the priority benefits of U.S. provisional application serial no. 63/376,068, filed on September 16, 2022. The entirety of the above-mentioned patent application is hereby incorporated by reference herein and made a part of this specification.

10 Technical Field

[0001] A point cloud is a collection of data points in a three-dimensional space. Point clouds are commonly used to represent the shape, appearance, and spatial distribution of objects or scenes in various applications. Point cloud coding, also known as point cloud compression, is essential due to many reasons, such as transmission efficiency, bandwidth constraints and so on. Specifically, point cloud data may be voluminous, especially when capturing detailed 3D scenes or objects. Efficient coding techniques are crucial for minimizing the data size of point cloud data, thus leading to accelerated transmission speeds and diminished storage space requirements.

15 [0002] Recently, Geometry Point Cloud Coding (G-PCC) is widely used in entertainment applications, e.g., VR/AR, and industrial applications, e.g., LiDAR sweep compression for automotive or robotics and HD map for navigation. MPEG released the first version G-PCC standard, and Audio Video Standard (AVS) is also developing a G-PCC standard. In order to compress the point cloud data efficiently, geometry information of a point cloud is compressed first, and then the corresponding attributes, including color or reflectance, are compressed based upon the geometry information. However, there are still many deficiencies that need to be improved in the existing G-PCC specification.

20

25

SUMMARY OF INVENTION

[Technical Problem]

[0003] The current AVS-GPCC specification cannot work well for a wide range of PCC input, and cannot be used in many applications.

5 [Solution to Problem]

[0004] This disclosure proposes several improvements for Geometry Point Cloud Coding. The proposed method may be used in future GPCC coding standards, in particular AVS-GPCC. With the implementation of the proposed method, modifications to bitstream structure, syntax, constraints, and mapping for the generation of decoded point cloud are considered for
10 standardizing.

[0005] The point cloud decoding method of the invention includes the following step: obtaining a bitstream of point cloud; and parsing a plurality of headers coded and point cloud data in layer structure in the bitstream of point cloud. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header. The sequence
15 header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

[0006] In an embodiment of the invention, the headers in the bitstream of point cloud further includes an attribute slice header and a geometry slice header, the geometry slice header is before
20 the attribute slice header. The step of parsing the headers and the point cloud data coded in layer structure in the bitstream of point cloud includes: when a syntax flag included in the frame header is equal to 1, parsing the attribute slice header and the geometry slice header for each slice of a PCC frame; and when a slice index is equal to 0 and the syntax flag included in the frame header is not equal to 1, parsing the attribute slice header and the geometry slice header for only one slice
25 of the PCC frame.

[0007] In an embodiment of the invention, the step of parsing the headers coded in layer

structure in the bitstream of point cloud includes: identifying the headers from the bitstream of point cloud according to a prefix bitstring and a predefined code. The last bit of the prefix bitstring is concatenated with the first bit of the predefined code.

[0008] In an embodiment of the invention, the prefix bitstring includes a 24-bit binary code, and
5 the predefined code includes an 8-bit binary code.

[0009] In an embodiment of the invention, the predefined code includes at least one of a slice start code, a slice end code, a pcc sequence start code, a pcc sequence end code, an attribute start code, a geometry start code, a frame start code, and a pcc edit code.

[0010] In an embodiment of the invention, the frame header includes a first syntax element
10 specifying a decoding order of a PCC frame. In another embodiment of the invention, the frame header includes a first syntax element specifying an order of a PCC frame associated with a plurality of point cloud frames. The first syntax element is configured to uniquely identify the associated PCC frame among all the point cloud frames. When the associated PCC frame is to
15 be output from a decoded point cloud buffer, the first syntax element is configured to indicate a position of the associated PCC frame in output order relative to output order positions of the other point cloud frames that are to be output from the decoded point cloud buffer.

[0011] In an embodiment of the invention, the frame header includes a second syntax element specifying a frame type of a PCC frame.

[0012] In an embodiment of the invention, the frame header includes a third syntax element
20 specifying whether there is more than one slice in a PCC frame.

[0013] In an embodiment of the invention, the frame header includes a third syntax element, a fourth syntax element, and a fifth syntax element. The third syntax element specifies whether there is more than one slice in a PCC frame, and the fourth syntax element specifies the number of slices in the PCC frame. The fifth syntax element specifies whether multiple geometry slice
25 headers and multiple attribute slice headers are used for the slices in the frame header when the third syntax element is equal to a preset value.

[0014] In an embodiment of the invention, a sixth syntax element specifying whether a current intra-coded PCC frame is a random access point and a seventh syntax element specifying whether a delta QP is enabled for a PCC frame are present in the frame header when a second syntax element specifying a frame type of the PCC frame is equal to a preset value.

5 [0015] In an embodiment of the invention, the method further includes: identifying a coding specification version of the bitstream of point cloud according to a syntax element associated with the coding specification version in at least one of the headers.

[0016] In an embodiment of the invention, the method further includes: obtaining a syntax element specifying the length of a sequence extension data and a syntax element specifying the
10 sequence extension data in at least one of the headers when the syntax element associated with the coding specification version is equal to a preset value.

[0017] In an embodiment of the invention, the method further includes: ignoring values of the sequence extension data by using a decoder conforming to a first coding specification version according to the syntax element specifying the length of the sequence extension data and the
15 syntax element specifying the sequence extension data when the syntax element associated with the coding specification version of the bitstream of point cloud is equal to the preset value. The coding specification version of the bitstream of point cloud is a second coding specification version when the syntax element associated with the coding specification version is equal to the preset value.

20 [0018] In an embodiment of the invention, an extension data and a user data are disposed after each of the headers, and the extension data and the user data are identified from the bitstream of point cloud according to a prefix bitstring and a predefined code.

[0019] In an embodiment of the invention, the prefix bitstring includes a 24-bit binary code, and the predefined code includes an 8-bit binary code.

25 [0020] In an embodiment of the invention, the predefined code includes an extension start code and a user data start code.

[0021] The point cloud encoding method of the invention includes the following step: obtaining point cloud data; and encoding the point cloud data to generate a bitstream of point cloud comprising a plurality of headers coded in layer structure. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header.

5 The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

[0022] In an embodiment of the invention, the headers in the bitstream of point cloud further includes an attribute slice header and a geometry slice header, and the geometry slice header is
10 before the attribute slice header. The step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer structure further includes: inserting the attribute slice header and the geometry slice header for each slice of a PCC frame into the bitstream of point cloud when a syntax flag included in the frame header is equal to 1; and inserting the attribute slice header and the geometry slice header for only one slice of the PCC frame into
15 the bitstream of point cloud when a slice index is equal to 0 and the syntax flag included in the frame header is not equal to 1.

[0023] In an embodiment of the invention, the step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer structure further includes: inserting a prefix bitstring and a predefined code into the bitstream of point cloud to indicate the
20 headers in the bitstream of point cloud, wherein the last bit of the prefix bitstring is concatenated with the first bit of the predefined code.

[0024] In an embodiment of the invention, the step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer structure further includes: inserting a syntax element according to a coding specification version of the bitstream of point
25 cloud into at least one of the headers in the bitstream of point cloud.

[0025] In an embodiment of the invention, the step of encoding the point cloud data to generate

the bitstream of point cloud comprising the headers coded in layer structure further includes: inserting a syntax element specifying the length of a sequence extension data and a syntax element specifying the sequence extension data into at least one of the headers when the syntax element associated with the coding specification version is equal to a preset value.

5 **[0026]** The decoder of the invention includes a storage device and a processor. The processor is electrically connected to the storage device, and is configured to: obtain a bitstream of point cloud; and parse a plurality of headers and point cloud data coded in layer structure in the bitstream of point cloud. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header. The sequence header is disposed
10 outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

[0027] The encoder of the invention includes a storage device and a processor. The processor is electrically connected to the storage device, and is configured to: obtain point cloud data; and
15 encode the point cloud data to generate a bitstream of point cloud comprising a plurality of headers coded in layer structure. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header. The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the
20 geometry header.

[Effects of Invention]

[0028] Based on the above, according to the point cloud decoding method, the point cloud encoding method, the decoder and the encoder of the invention, the code stream can be efficiently decoded, and the data of the point cloud can be efficiently encoded.

25 **[0029]** To make the aforementioned more comprehensible, several embodiments accompanied with drawings are described in detail as follows.

BRIEF DESCRIPTION OF DRAWINGS

[0030] FIG. 1 is a schematic diagram of a G-PCC encoding process according to an embodiment of the invention.

5 [0031] FIG. 2A and FIG. 2B illustrate octree structures of G-PCC according to an embodiment of the invention.

[0032] FIG. 2C illustrates corresponding digital representation of the octree structure according to an embodiment of the invention.

[0033] FIG. 3 illustrates structure of the cubes according to an embodiment of the invention.

10 [0034] FIG. 4 is a schematic diagram of a G-PCC decoding process according to an embodiment of the invention.

[0035] FIG. 5A and 5B are schematic diagrams of a syntax table of color residual coding according to an embodiment of the invention.

15 [0036] FIG. 6 is a schematic diagram of a syntax table of reflectance coding according to an embodiment of the invention.

[0037] FIG. 7 is a schematic diagram of a syntax table of zero-run length coding according to an embodiment of the invention.

[0038] FIG. 8 is a flowchart of a point cloud decoding method applied to a decoder according to an embodiment of the invention.

20 [0039] FIG. 9 is a flowchart of parsing the attribute slice header and the geometry slice header according to an embodiment of the invention.

[0040] FIG. 10A is a schematic diagram of a syntax table of a bitstream of point cloud according to an embodiment of the invention.

25 [0041] FIG. 10B is a schematic diagram of a syntax table of a geometry slice according to an embodiment of the invention.

[0042] FIG. 10C is a schematic diagram of a syntax table of an attribute slice according to an

embodiment of the invention.

[0043] FIG. 11 is a flowchart of identifying the headers according to an embodiment of the invention.

5 [0044] FIG. 12 is a schematic diagram of a syntax table of a frame header according to an embodiment of the invention.

[0045] FIG. 13 is a flowchart of recognizing a coding specification version of the bitstream of point cloud according to an embodiment of the invention.

[0046] FIG. 14A and 14B are schematic diagrams of a syntax table of a sequence header according to an embodiment of the invention.

10 [0047] FIG. 15A and 15B are schematic diagrams of a syntax table of a geometry header according to an embodiment of the invention.

[0048] FIG. 16A and 16B are schematic diagrams of a syntax table of an attribute header according to an embodiment of the invention.

15 [0049] FIG. 17A and 17B are schematic diagrams of a syntax table of a geometry slice header according to an embodiment of the invention.

[0050] FIG. 18A and 18B are schematic diagrams of a syntax table of an attribute slice header according to an embodiment of the invention.

[0051] FIG. 19 is a schematic diagram of a syntax table of a sequence header according to an embodiment of the invention.

20 [0052] FIG. 20 is a schematic diagram of an extension and user data structure according to an embodiment of the invention.

[0053] FIG. 21A and FIG. 21B are schematic diagrams of an extension data structure according to an embodiment of the invention.

25 [0054] FIG. 22 is a schematic diagram of an user data structure according to an embodiment of the invention.

[0055] FIG. 23 is a flowchart of a point cloud encoding method applied to an encoder according

to an embodiment of the invention.

[0056] FIG. 24 is a schematic block diagram of an codec device provided by an embodiment of the invention.

5

DESCRIPTION OF EMBODIMENTS

[0057] In order to have a more detailed understanding of the characteristics and technical content of the embodiments of the invention, the implementation of the embodiments of the invention will be described in detail below with reference to the accompanying drawings. The attached drawings are for reference and explanation purposes only, and are not used to limit the
10 embodiments of the invention.

[0058] This disclosure proposes a coding method for G-PCC. The proposed method may be used in future G-PCC coding standards, such as AVS-GPCC. With the implementation of the proposed method, modifications to bitstream structure, syntax, constraints, and mapping for the generation of decoded point cloud are considered for standardizing.

15 [0059] Point cloud refers to a set of discrete points in space with irregular distribution, representing the spatial structure and surface attributes of a three-dimensional object or scene. Each point in the point cloud can include the position information of the point and the attribute information of the point. The position information of a point may be three-dimensional coordinate information of a point. The position information of a point can also be referred to as
20 the geometric information of a point. On the other hand, the attribute information of a point may be color information and/or reflectivity.

[0060] As an example, the color information of a point can encompass data from various color spaces, such as the RGB color space, YCbCr color space, or YUV color space. That is, the color information may include RGB information. Alternatively, the color information may include
25 luminance information and chromaticity information. Besides, based on the principles of laser measurement, the points in the point cloud can include the three-dimensional coordinate

information of the points and the laser reflectance intensity (reflectance). As another example, based on the principles of photogrammetry measurement, the points in the point cloud can include the three-dimensional coordinate information of the points and the color information.

[0061] Point cloud data may be acquired through various methods, which include but are not restricted to, at least one of the following: (1) computer equipment generation, wherein computer equipment can generate point cloud data based on virtual 3D objects and virtual 3D scenes; (2) 3D laser scanning acquisition, wherein through 3D laser scanning, point cloud data of static real-world three-dimensional objects or three-dimensional scenes can be obtained, and millions of point cloud data can be obtained every second; (3) 3D photogrammetry acquisition, wherein the point cloud data of the real-world visual scene can be obtained by 3D photography equipment (that is, a group of cameras or camera equipment with multiple lenses and sensors) to obtain the point cloud data of the real-world visual scene, and the point cloud data of dynamic real-world three-dimensional objects or three-dimensional scenes can be obtained through 3D photography; and (4) Obtain point cloud data of biological tissues and organs through medical equipment. In the medical field, point cloud data of biological tissues and organs can be obtained through medical equipment such as magnetic resonance imaging (MRI), computed tomography (CT), and electromagnetic positioning information.

[0062] These technologies have reduced the cost and time cycle of point cloud data acquisition meanwhile enhancing data accuracy. However, the processing of massive 3D point cloud data may face challenge of limitations in storage space and transmission bandwidth. G-PCC is one solution to overcome such issue. G-PCC generally includes the compression of geometry information and the compression of attribute information, and the operations of point cloud coding is introduced below.

[0063] FIG. 1 is a schematic diagram of a G-PCC encoding process according to an embodiment of the invention. Referring to FIG. 1, the G-PCC encoding process shown in FIG. 1 may be implemented by a point cloud encoder. For the point cloud data to be encoded, the point cloud

data is divided into multiple slices through slice division. In each slice, the positions (i.e., geometric information) of the points and the attributes of the points are coded (i.e., encoded and decoded) separately. From the above, it can be understood that points in a point cloud can have both position information and attribute information. Therefore, the encoding of points may involve position encoding and attribute encoding. In some cases, the position information of points is referred to as geometric information, and the encoding of point position can also be termed as geometric encoding.

[0064] In the geometric encoding process, in step S102, a transform to the coordinates of the points is performed to transform the coordinates of the points from an initial domain to a transform domain. In particular, the positions of the points are coordinated to enclose the point cloud into a bounding box (bounding box). In step S104, the transform coordinates are voxelized. Voxelization of the transform coordinates may include quantization and removing some points of the point cloud. The quantization mainly plays the role of scaling. Due to the quantization rounding, part of the positions of the point cloud is the same, so whether to remove the duplicate points based on the parameters is further determined in step S104. The process of quantifying and removing the duplicate points is also called the voxelization process.

[0065] In step S106, an octree is generated based on the voxelized transform coordinates. In other words, the bounding box is divided into an octree for octree analysis. In the octree-based geometric information encoding process, the bounding box is divided into eight sub-cubes, and the non-empty (including points in the point cloud) sub-cubes are continued to be divided into eight equal parts until the leaf nodes are obtained. In step S108, the points in the leaf nodes of the octree structure are arithmetic coded to generate a binary geometric bitstream, that is, a geometric code stream.

[0066] FIG. 2A and FIG. 2B illustrate octree structures of G-PCC according to an embodiment of the invention, and FIG. 2C illustrates corresponding digital representation of the octree structure according to an embodiment of the invention. Referring to FIG. 2A, a cubical axis-aligned

bounding box B is defined by two extreme points $(0,0,0)$ and $(2^d, 2^d, 2^d)$ where d is the maximum size of a given point cloud along x, y or z direction. A point of the point cloud will be noted as “point” below. All points are included in this defined cube B.

[0067] Referring to FIG. 2B, the cube B is divided into eight sub-cubes B1 to B8, which creates an octree structure allowing one parent cube B to have 8 child cubes B1 to B8. The 7 sibling cubes B2 to B8 of a given cube B1 are the same size cubes and share at least one same face/edge/point with this given cube. The volume of each of the cubes B1 to B8 is 1/8 volume of its parent cube B. Each of the cubes B1 to B8 may contain more than one point, and a number of points in a cube is dependent on the size and location of the cube. The size of a smallest cube is pre-defined for a given point cloud. For a given point, the parent cube of a given point is defined as a minimum size cube which contains this given point. Sibling points of a given point are defined as those points which have the same parent cube with this given point.

[0068] Referring to FIG. 2C, an octree is a recursive data structure that is often used to describe three-dimensional space in which each internal cube has exactly eight children. The space is recursively subdivided into eight octants to the point where the resolution of the child cube is equal to a size of the point – the smallest element that has no further subdivisions. To represent a cube an 8-bit binary code that follows a space-filling curve pattern (Hilbert, Morton) is used, each child is assigned a “1” or “0” value to indicate if the space in the child cube has any points associated with that child cube, or the child cube is empty. Only the occupied child cubes are further subdivided. The process of parent cube subdivision is terminated when the size of the child cube becomes equal to the size of the indivisible element, i.e., spatial resolution of the point cloud, or simply the size of the point.

[0069] FIG. 3 illustrates structure of the cubes according to an embodiment of the invention. Referring to FIG. 3, depending on the location of the current cube, one cube may have up to six same-size cubes to share one face. In addition, the current cube may also have some neighbouring cubes which share lines or points with the current cube.

[0070] Similarly, the parent cube of the current cube also has up to six neighbouring cubes with the same size of the parent cube that share one face with the parent cube. The parent cube of the current cube also has up to twelve neighbouring cubes with the same size of parent cubes that share an edge. The parent cube of the current cube also has up to eight neighbouring cubes with the same size of parent cubes that share a point with the parent cube.

[0071] Referring back to FIG. 1, in step S110, the points is analyzed to potentially determine a surface representation of sets of the points. In other words, based on the surface formed by the distribution of the point cloud in each block, twelve pieces of the surface and the block are analyzed. At most twelve vertexes (intersection points) are generated by the edges, and the arithmetic coding is performed on the vertexes (surface fitting based on the intersection points) in step S108 to generate the binary geometric bitstream. In step S112, the transform coordinates of points in the point cloud may be reconstructed based on the octree, data indicating the surfaces determined in step S110 and/or other information. The reconstructed geometric information may be used by attribute coding process.

[0072] In the attribute coding process, after the geometric information is reconstructed, color transformation is performed in step S114, in which the color information (that is, the attribute information) is transformed from the RGB color space to the YUV color space, YCbCr color space or other color space. In step S116, the reconstructed geometric information is used to recolor the point cloud, so that the uncoded attribute information corresponds to the reconstructed geometric information. Because the geometric information changes after the geometric encoding, it is necessary to reassign the attribute value to each point after the geometric encoding.

[0073] In the process of color information coding, there are mainly two transformation methods. The first method is distance-based lifting transformation in step S120 that relies on Level of Detail (LOD) division in step S118. The second method is Region Adaptive Hierarchical Transform (RAHT) transformation in step S122, which directly transforms the image. Both of these methods convert color information from the spatial domain to the frequency domain, to obtain

high-frequency coefficients and low-frequency coefficients through the transformation. The high-frequency coefficients and low-frequency coefficients may be referred to transform coefficients. Next, in step S124, these transform coefficients are quantified to generate quantized transform coefficients. In some case, quantizing the transform coefficients for a point may involve dividing the quantized attribute value (i.e., quantized transform coefficients) by a QP step size for the point, and the QP step size is determined according to a Quantization Parameter (QP). In step S126, arithmetic encoding may be performed to syntax elements representing the quantized attribute coefficients, to output these syntax elements in an attribute bitstream. Namely, the quantized transform coefficients may be encoded by using arithmetic encoding to generate a binary attribute bitstream, that is, the attribute code stream.

[0074] FIG. 4 is a schematic diagram of a G-PCC decoding process according to an embodiment of the invention. The process in FIG. 4 may be implemented by the point cloud decoder. For the obtained binary code stream, the geometric bitstream and the attribute bitstream in the binary code stream are first decoded in steps S402 and S404, respectively. When decoding the geometric bitstream, through arithmetic decoding in step S402, octree synthesis in step S406, surface fitting in step S408, geometry reconstruction in step S410, and inverse coordinate transformation in step S412, the positions (i.e., geometric information) of the point cloud are obtained.

[0075] When decoding the attribute bitstream, through arithmetic decoding in step S404, inverse Quantization in step S414, LOD-based lifting based inverse transformation in steps S416 and S418, or RAHT based inverse transformation in step S420, and inverse color conversion in step S422, the attributes of the point cloud are obtained. It should be noted that, the quantized attribute values (i.e., quantized transform coefficients) may be dequantized in step S414. In some case, dequantizing a quantized attribute value for a point may involve multiplying the quantized attribute value by a QP step size for the point, and the QP step size is determined according to a Quantization Parameter (QP).

[0076] The octree-based geometry information may be coded with context-based arithmetic coding. There may also be some corresponding attribute information for point clouds, including color, reflectance, etc., that needs to be compressed. Because the neighbouring points in a point cloud may have a strong correlation, prediction-based coding methods have been developed and used to compose and code point cloud attributes. More specifically, a prediction is formed from neighbouring coded attributes. Then, the difference between the current attribute and the prediction is coded.

[0077] It should be noted that, the steps depicted in FIG. 1 and FIG. 4 may be implemented as a plurality of program codes or circuits, which are not particularly limited in the invention.

[0078] In the disclosure, coding may be assumed to mean encoding and decoding methods and systems.

[0079] [Attribute Coding in the AVS G-PCC]

[0080] AVS is developing a G-PCC standard. After the geometry information is coded, Morton or Hilbert code/order may be used to convert a point cloud cube into a one-dimension array. Each position in the cube will have a corresponding Morton or Hilbert code, but some positions may not have any corresponding point cloud attribute. In other words, some positions may be empty. The attribute coding will follow the pre-defined Morton or Hilbert order. A predictor may be generated from the previous coded points in Morton or Hilbert order. The attribute difference between the current point and its predictor is coded into the bitstream.

[0081] To reduce the memory usage, some pre-defined number has been specified to limit the number of neighboring points that can be used in generating the prediction. For example, only M data points among previous N consecutively coded points may be used for coding the current attribute. In the previous AVS G-PCC software, M and N are set as a fixed number of 3 and 128, respectively.

[0082] If more than 128 points have already been coded before the current point, only 3 out of the 128 previously coded neighbouring points could be used to form the attribute predictor

according to a pre-defined order. If there are less than 128 coded points before the current point, all such coded points will be used as candidates to establish the attribute predictor. More specifically, the previous K points, e.g., K = 6, before the current point are selected according to a pre-defined Morton or Hilbert order. Then, new Morton or Hilbert codes for these N points will be re-calculated by adding a fixed shift, e.g., 1, to coordinates (x, y, z) of these N data points. Assuming that the new Morton or Hilbert code for the current position is X, a P-point set before and a Q-point set after the current position according to the new Morton or Hilbert code order will be selected. Among these pre-defined K, P and Q point sets, M points are selected with M closest “distance” between these coded points and the current point. The distance $d1$ as one example is defined as follows, other distance metrics can also be used.

$$[0083] \quad d1 = |x1 - x2| + |y1 - y2| + |z1 - z2| \quad (1)$$

[0084] where $(x1, y1, z1)$ and $(x2, y2, z2)$ are coordinates of the current point and pre-selected point, respectively.

[0085] More recently, a full search method based on Hilbert code has been applied in the AVS GPCC attribute coding. In the current software, the search range is set as 128 and the number of previous points used to form the predictor is set as M. If more than 128 points have already been coded before the current point, only M out of the 128 previously coded neighbouring points can be used to form the attribute predictor according to the Hilbert order.

[0086] If there are fewer than 128 coded points before the current one, all such coded points will be used as candidates to form the attribute predictor. Among the up-to 128 previously coded points, M points are selected with M closest “distance” between these coded points and the current point. The distance $d2$ as one example is defined as follows, other distance metrics can also be used.

$$[0087] \quad d2 = |x1 - x3| + |y1 - y3| + |z1 - z3| \quad (2)$$

[0088] where $(x1, y1, z1)$ and $(x3, y3, z3)$ are the coordinates of the current point and the pre-selected point along Hilbert order, respectively. Once M closest points have been selected,

a weighted average of attributes from these M points is formed as the predictor to code the attribute of the current point.

[0089] It is known that the points which share the same face/line/point with the current point are close to the current point. Another technology is to consider these points serving as a
5 predictor.

[0090] The residual is defined as the difference of attribute values between the current point and its predictor. Depending on the application, PCC can be either lossless or lossy. Hence, the residual may or may not be quantized by using the predefined quantization process. In the disclosure, the residual without or with quantization is called level. The level can be a signed
10 integer and will be coded into the bitstream.

[0091] [Color Level Coding]

[0092] There are three color attributes for each point, which come from the three color components. If the levels for all the three color components are zeros, this point is called a zero level point. Otherwise, if there is at least one non-zero level for one color component with the
15 point, this point is called a non-zero level point. In the current AVS-GPCC, the number of consecutive zero level points is called zero-run length. The zero-run length values and levels for non-zero level points are coded into the bitstream.

[0093] More specifically, on the encoding side, before coding the first point, the zero-run length value is set as zero. Starting from the first point along the predefined coding order, the residuals
20 between the three color predictors and their corresponding color attributes for the current point can be obtained. Then, the corresponding levels for the three components of the current point also can be obtained. If the current point is a zero level point, the zero-run length value will be increased by one and the process proceeds to the next point. If the current point is a non-zero level point, the zero-run length value will be coded first and then the three color levels for this
25 non-zero level point will be coded right after. After the level coding of a non-zero level point, the zero-run length value will be reset to zero and the process proceeds to the next point till

finishing all points.

[0094] On the decoding side, the zero-run length value is first decoded and the three color levels corresponding to the number of zero-run length points are set as zero. Then, the levels for the non-zero level point are decoded and then the next zero-run length value is decoded. This process continues until all points are decoded.

[0095] For example, referring to FIG. 5A and 5B, which are schematic diagrams of a syntax table 500a and 500b of color residual coding according to an embodiment of the invention. For a non-zero level point, there is at least one non-zero level among the three components. The values of three components are coded in the `color_residual_coding()`. Several one-bit flags plus the remainder of the absolute level may be coded to represent levels of three components. The absolute level or absolute level of color residual minus one may be coded in the function `coded_level_coding()` and named as coded level in the remaining of this disclosure.

[0096] In one exemplary case, a first flag (e.g., `color_first_comp_zero`) is coded to indicate if the first component of color is zero or not; if the first component of color is zero, a second flag (e.g., `color_second_comp_zero`) is coded to indicate if the second component of color is zero or not; if the second component of color is zero, the absolute level minus one and the sign of the third component will be coded according to the following coded level method.

[0097] In another case, a first flag is coded to indicate if the first component of color is zero or not; if the first component of color is zero, a second flag is coded to indicate if the second component of color is zero or not; if the second component of color is not zero, the absolute level minus one and sign of the second component will be coded according to the following coded level method, the absolute level and sign of the third component will be coded according to the following coded level method.

[0098] In another case, a first flag is coded to indicate if the first component of color is zero or not; if the first component of color is not zero, the absolute level minus one and sign of the first component will be coded according to the following coded level method, the absolute levels and

signs of the second and third components will be coded according to the following coded level method, respectively.

[0099] The coding method to produce a coded level is designed as follows. The first flag (e.g., coded_level_equal_zero) is coded to indicate if the coded level is zero or not; if the coded level is the absolute level of one component minus one, i.e., when isComponentNoneZero is equal to true, the sign (e.g., coded_level_sign) of the level of this component will be coded. If the first flag indicates that the coded level is not zero, and if the coded level is the absolute level of one component itself, i.e., when isComponentNoneZero is equal to false, the sign of the level of this component will be coded. The second flag (e.g., coded_level_gt1) will be coded to indicate if the coded level is greater than one or not; if the coded level is greater than one, the parity of the coded level minus two is coded and the third flag (e.g., coded_level_minus2_div2_gt0) will be coded to indicate if the coded level minus two dividing two is greater than zero or not; if the coded level minus two dividing two is greater than zero, the coded level minus two dividing two minus one will be coded.

[00100] [Reflectance Level Coding]

[00101] In the current AVS-GPCC specification, the zero-run length of reflectance level and the non-zero reflectance level are coded into bitstream. More specifically, in the encoding side, before coding of the first point, the zero-run length counter is set as zero. Starting from the first point along the predefined coding order, the residuals between the predictors and corresponding original points are obtained. Then the corresponding levels are obtained. If current level is zero, zero-run length will be increased by one and the process proceeds to the next point. If the level is not zero, the zero-run length will be coded first and the non-zero level will be coded after the zero-run length coding. After coding a non-zero level, zero-run length counter will be reset to zero and the process proceeds to next point. In the decoding side, the number of zero-run length is first decoded and the levels corresponding to the number of zero-run length points are set as zero. Then non-zero level is decoded and the next number of zero-run length is decoded. Process

continues until all points are decoded.

[00102] For example, referring to FIG. 6, which is schematic diagram of a syntax table 600 of reflectance coding according to an embodiment of the invention. For a non-zero level, if the current point is not a duplicated point, the sign of level is coded with a syntax element residual_sign first. Then a syntax element abs_level_minus1_parity which indicates the parity of the absolute level minus one is coded. Another syntax element abs_level_minus1_div2_gt0 is coded to indicate if the value of the absolute level minus one divided by two is greater than zero or not; if abs_level_minus1_div2_gt0 is great than zero, another syntax element abs_level_minus1_div2_gt1 is coded to indicate if the value of the absolute level minus one divided by two is greater than one or not; if abs_level_minus1_div2_gt1 is greater than 1, another syntax element abs_level_minus1_div2_minus2 is coded to indicate the value of the absolute level minus one divided by two minus two.

[00103] [Zero-run length coding]

[00104] In the current AVS-GPCC specification, the value of the zero-run length is coded into bitstream. For example, referring to FIG. 7, which is a schematic diagram of a syntax table 700 of zero-run length coding according to an embodiment of the invention. More specifically, the first syntax zero_run_length_level_equal_zero is coded to indicate if the zero-run length is equal to zero or not; if it is not zero, the second syntax zero_run_length_level_equal_one is coded to indicate if the zero-run length is equal to one or not; if it is not one, the third syntax zero_run_length_level_equal_two is coded to indicate if the zero-run length is equal to two or not; if it is not two, the fourth syntax zero_run_length_level_minus3_parity and the fifth syntax zero_run_length_level_minus3_div2 are coded to indicate the parity of the zero-run length minus three and the value of the zero-run length minus three divided by two, respectively.

[00105] [Aspect 1: bitstream structure]

[00106] FIG. 8 is a flowchart of a point cloud decoding method applied to a decoder according to an embodiment of the invention. Referring to FIG. 8, each step in the methods may be

performed by a decoder using the G-PCC decoding process illustrated as FIG. 4. In step S810, the decoder may obtain a bitstream of point cloud. In step S820, the decoder may parse a plurality of headers and point cloud data coded in layer structure in the bitstream of point cloud. The headers in the bitstream of point cloud includes at least a sequence header, a geometry header, an attribute header, and a frame header, and other headers. The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header. That is, the headers are disposed in a specific order within the bitstream. Each header may include information (e.g., parameters, identifiers, or flags) for decoding the bitstream of point cloud. Besides, the point cloud data in the bitstream of point cloud may further include geometry data and attribute data. In some embodiments, the geometry data may include general geometry data bitstream of one or more slice(s) of a PCC frame. In some embodiments, the attribute data may include the general attribute data bitstream of one or more slice(s) of a PCC frame. For example, the geometry bitstream may be decoded according to the information in the geometry header. The attribute bitstream may be decoded according to the information in the attribute header. However, when a syntax element, which is included in the sequence header and specifying whether the attribute data is present, is equal to 0, the attribute header and the attribute data may not be included in the bitstream of point cloud.

[00107] In an embodiment of the invention, the headers in the bitstream of point cloud may further include an attribute slice header and a geometry slice header, and the geometry slice header is before the attribute slice header. When a syntax flag included in the frame header is equal to 1 and a syntax element included in the sequence header indicates the attribute data is present, the decoder may parse the attribute slice header and the geometry slice header for each slice of a PCC frame. When a syntax flag included in the frame header is equal to 1 and a syntax element included in the sequence header indicates the attribute data is not present, the decoder may parse the geometry slice header for each slice of a PCC frame. When a slice index is equal to 0, the

syntax flag included in the frame header is not equal to 1 (i.e., the syntax flag is equal to 0), and a syntax element included in the sequence header indicates the attribute data is present, the decoder may parse the attribute slice header and the geometry slice header for only one slice of the PCC frame. When a slice index is equal to 0, the syntax flag included in the frame header is not equal to 1 (i.e., the syntax flag is equal to 0), and a syntax element included in the sequence header indicates the attribute data is not present, the decoder may parse the geometry slice header for only one slice of the PCC frame. Further, when a slice index is not equal to 0 and the syntax flag included in the frame header is not equal to 1 (i.e., the syntax flag is equal to 0), the decoder may not parse the attribute slice header and the geometry slice header for the current slice of the PCC frame.

[00108] FIG. 9 is a flowchart of parsing the attribute slice header and the geometry slice header according to an embodiment of the invention. Referring to FIG. 9, each step in the methods may be performed by a decoder using the G-PCC decoding process illustrated as FIG. 4. In step S821, the decoder may determine whether a syntax flag included in the frame header is equal to 1. If the step S821 is determined as yes, in step S822, the decoder may parse the attribute slice header and the geometry slice header for each slice of a PCC frame when a syntax element included in the sequence header indicates the attribute data is not present. That is, when the syntax flag included in the frame header is equal to 1, each slice may be corresponding to one attribute slice header and one geometry slice header, and the bitstream may include multiple attribute slice headers and multiple geometry slice headers for multiple slices in the same PCC frame. On the other hand, if the step S821 is determined as no, in step S823, the decoder may parse the attribute slice header and the geometry slice header for only one slice of the PCC frame when the slice index is equal to 0 when a syntax element included in the sequence header indicates the attribute data is not present. That is, in step S823, the bitstream may only include one attribute slice header and one geometry slice header for all slices in the same PCC frame.

[00109] FIG. 10A is a schematic diagram of a syntax table 100 of a bitstream of point cloud

according to an embodiment of the invention. FIG. 10B is a schematic diagram of a syntax table of a geometry slice according to an embodiment of the invention. FIG. 10C is a schematic diagram of a syntax table of an attribute slice according to an embodiment of the invention. This disclosure proposes that the bitstream of point cloud is organized with the layered structures in

5 FIG. 10A to FIG.10C. The sequence header is the first parsed structure which includes some sequence related syntax. After the first sequence header, one geometry header may be parsed. Then, if a syntax element (i.e., **attribute_present_flag**) specifying whether the attribute is present is equal to 1, one attribute header may be parsed after parsing the geometry header. If a syntax element (i.e., **attribute_present_flag**) specifying whether the attribute is present is equal to 0, one

10 attribute header may not be parsed after parsing the geometry header. The bitstream starts with the first sequence header and ends with either the first sequence ending code or the first editing code. It should be noted that, FIG. 10A merely takes one sequence header, one geometry header and one attribute header for example, the number of the sequence header, the number of the geometry header and the number of attribute header is not limited in the disclosure. That is, there

15 may be multiple sequence headers, geometry headers and attribute headers between the first sequence header among the multiple sequence headers and the first sequence ending code or the first editing code.

[00110] In an embodiment of the invention, except for that the first sequence header, the first geometry header and the first attribute header may appear in order, the other sequences headers,

20 the other geometry headers, and the other attribute headers may appear one, two, or three of them and in any order.

[00111] In an embodiment of the invention, after the sequence header, the geometry header and the attribute header, there may be one or multiple coded PCC frames. In the exemplary case of FIG. 10A, there is a PCC frame header associated with each coded PCC frame. The latest

25 sequence header, the latest geometry header, and the latest attribute header maybe used for coding the current PCC frame. Coded bitstream has same decoding order with coded PCC frame. The

coding order of PCC frame may be different with display order of PCC frame.

[00112] In an embodiment of the invention, a syntax table of a bitstream of point cloud may include a conditioning statement determining whether to parse the attribute slice header and the geometry slice header, wherein the conditioning statement may include checking whether the slice index is equal to 0 and whether a syntax flag (e.g., **multi_slice_header_enabled_flag** in FIG. 12) included in the frame header is equal to 1. If the slice index is equal to 0 or the syntax flag (e.g., **multi_slice_header_enabled_flag** in FIG. 12) is equal to 1, the decoder may parse the attribute slice header and the geometry slice header when a syntax element (i.e., **attribute_present_flag**) is equal to 1. If the slice index is equal to 0 or the syntax flag (e.g., **multi_slice_header_enabled**) is equal to 1, the decoder may parse the geometry slice header when a syntax element (i.e., **attribute_present_flag**) is equal to 0. Otherwise, if the slice index is not equal to 0 and the syntax flag (e.g., **multi_slice_header_enabled_flag** in FIG. 12) is not equal to 1, the decoder may not parse the attribute slice header and the geometry slice header for the current slice.

[00113] In an embodiment of the invention, the decoder may identify the headers from the bitstream of point cloud according to a prefix bitstring and a predefined code. The last bit of the prefix bitstring is concatenated with the first bit of the predefined code. Namely, a code consisting of the prefix bitstring and the predefined code may be used to identify the header and data information.

[00114] In an embodiment of the invention, the prefix bitstring includes 24-bit binary code, and the predefined code includes 8-bit binary code. For example, the prefix bitstring may be 24-bit binary code "0000 0000 0000 0000 0000 0001".

[00115] In an embodiment of the invention, the predefined code includes at least one of a slice start code, a slice end code, a pcc sequence start code, a pcc sequence end code, an attribute start code, a geometry start code, a frame start code, and a pcc edit code. The value of the predefined code may be shown in Table 1.

Table 1

Type of predefined code	Value
slice_start_code	00~7F
reserved	80~8E
slice_end_code	8F
reserved	90~AF
pcc_sequence_start_code	B0
pcc_sequence_end_code	B1
attribute_start_code	B2
geometry_start_code	B3
frame_start_code	B4
user_data_start_code	B5
pcc_edit_code	B6
extension_start_code	B7
reserved	B8
System codes	B9~FF

[00116] FIG. 11 is a flowchart of identifying the headers according to an embodiment of the invention. Referring to FIG. 11, each step in the methods may be performed by a decoder using the G-PCC decoding process illustrated as FIG. 4. In step S825, the decoder may determine whether the next sequence of bits match to a prefix bitstring. If the step S825 is determined as yes, the decoder may identify the headers from the bitstream of point cloud according to a prefix bitstring and a predefined code.

[00117] For example, based on the Table 1 and syntax table 100 in FIG. 10, if the next 32 bits in the bitstream is “0000 0000 0000 0000 0000 0001” concatenated with the “1011 0000”, the decoder may be able to identify the pcc_sequence_start_code which is used for identifying the sequence header.

[00118] [Aspect 2: AVS_GPCC frame header]

[00119] This disclosure proposes a frame header for a PCC frame. All the frame-related syntax elements (e.g., frame number, frame coding type, etc.) are specified in the frame header.

[00120] In an embodiment of the invention, the frame header includes a first syntax element

specifying a decoding order of a PCC frame. In another embodiment of the invention, the frame header includes a first syntax element specifying an order of a PCC frame associated with a plurality of point cloud frames. The first syntax element is configured to uniquely identify the associated PCC frame among all the point cloud frames. When the associated PCC frame is to be output from a decoded point cloud buffer, the first syntax element is configured to indicate a position of the associated point cloud frame in output order relative to output order positions of the other point cloud frames that are to be output from the decoded point cloud buffer.

[00121] In an embodiment of the invention, the frame header includes a second syntax element specifying a frame type of a PCC frame.

[00122] In an embodiment of the invention, the frame header includes a third syntax element specifying whether there is more than one slice in a PCC frame.

[00123] In an embodiment of the invention, the frame header includes a third syntax element specifying whether there is more than one slice in a PCC frame. A fourth syntax element specifying the number of slices in the PCC frame and a fifth syntax element specifying whether multiple geometry slice headers and multiple attribute slice headers are used for the slices are present in the frame header when the third syntax element is equal to a preset value.

[00124] In an embodiment of the invention, a sixth syntax element specifying whether a current intra-coded PCC frame is a random access point and a seventh syntax element specifying whether a delta QP is enabled for a PCC frame are present in the frame header when a second syntax element specifying a frame type of the PCC frame is equal to a preset value.

[00125] For example, FIG. 12 is a schematic diagram of a syntax table 200 of a frame header according to an embodiment of the invention. Referring to FIG. 12, the frame header may include a first syntax element (i.e., **decode_order_index**) specifying a decoding order of a PCC frame. In another example of the invention, the frame header may include a first syntax element (i.e., **decode_order_index**) specifying an order of a PCC frame that is associated with each point cloud frame. The first syntax element (i.e., **decode_order_index**) may be configured for

uniquely identifying the associated PCC frame among all the point cloud frames. When the associated PCC frame is to be output from the decoded point cloud buffer, the he first syntax element (i.e., **decode_order_index**) may be configured for indicating the position of the associated PCC frame in output order relative to the output order positions of the other point cloud frames that are to be output from the decoded point cloud buffer.

[00126] The frame header may include a second syntax element (i.e., **frame_type**) specifying a frame type of a PCC frame which is specified in Table 2.

Table 2

Codes in binary	The frame coding type
00	Intra PCC
01	P PCC
10	B PCC
11	Reserved

[00127] Besides, the third syntax element (i.e., **mul_slice_flag**) in the frame header specifying whether there is more than one slice in a PCC frame. In one exemplary case, **mul_slice_flag** equal to 0 specifies that there is only one slice within the current picture, and **mul_slice_flag** equal to 1 specifies that there is more than one slice within the current picture.

[00128] A fourth syntax element (i.e., **num_slice_minus_two**) specifying the number of slices in the PCC frame and a fifth syntax element (i.e., **mult_slice_header_enabled_flag**) specifying whether multiple geometry slice headers and multiple attribute slice headers are used for the slices are present in the frame header when the third syntax element (i.e., **mul_slice_flag**) is equal to 1.

In detail, **num_slice_minus_two** plus 2 specifies the number of slices for the current picture.

When **num_slice_minus_two** is not present, it is inferred to be equal to -1. The number of slices (referred to num_slice) may be obtained according to the following formula (1).

$$\text{num_slice} = \text{num_slice_minus_two} + 2 \quad \text{formula (1)}$$

Further, **mult_slice_header_enabled_flag** equal to 1 specifies that multiple geometry slice

headers and the attribute slice headers may be used for multiple slices, and **mult_slice_header_enabled_flag** equal to 0 specifies that only one geometry slice header and one attribute slice header are used for the current frame. When **mult_slice_header_enabled_flag** is not present, it is inferred to be equal to 0.

5 [00129] In an embodiment of the invention, a sixth syntax element (i.e., **random_access_point_flag**) specifying whether a current intra-coded PCC frame is a random access point and a seventh syntax element (i.e., **frame_qp_delta_enabled_flag**) specifying whether a delta QP is enabled for a PCC frame are present in the frame header when a second syntax element specifying a frame type of the PCC frame is equal to “00”. In one exemplary
10 case, **random_access_point_flag** equal to 1 specifies that the current intra-coded PCC frame may be a random access point, and **random_access_point_flag** equal to 0 specifies that the current intra coded PCC frame may not be a random access point. When **random_access_point_flag** is not present, it is inferred to be 0. Further, **pic_qp_delta_enabled_flag** equal to 1 specifies that delta quantization parameter (QP) may be enabled for the current PCC frame, and
15 **pic_qp_delta_enabled_flag** equal to 0 specifies that delta QP may not be enabled for the current PCC frame.

[00130] [Aspect 3: The extension of AVS-GPCC]

[00131] To add more coding tools in future coding specification versions, it is desirable to design the interface supporting compatibility between the current standard and extension standards.
20 First, the later version standard (e.g., GPCC v2) decoder can fully decode the previous version standard (e.g., GPCC v1) bitstreams, i.e. backward compatibility. Second, it is better for a previous standard (e.g., GPCC v1) decoder can also be forward compatible at least with the high level syntax portion of bitstreams produced by later versions of GPCC (e.g., GPCC v2).

[00132] More specifically, a decoder conforming to GPCC v1 may not be able to decode the
25 entirety of a GPCC v2 bitstream of, because the GPCC v2 bitstream may exercise coding tools which are defined in the GPCC v2 specification but are not known to GPCC v1 decoders.

However, it is highly desirable that a GPCC v1 decoder should at least be able to decode the high level syntax portion of a GPCC v2 bitstream. By decoding the high level syntax successfully, the GPCC decoder may determine not just tool information, but also profile and tier information. Such information provides hints to the decoder of the capabilities required to decode the bitstream.

5 For example, tool information hints to the decoder which coding tools are used in the bitstream. The profile and tier information hints to the decoder about the uncompressed video data throughput that needs to be supported.

[00133] If the high level syntax is successfully parsed, the GPCC decoder can determine if the current bitstream can be decoded. If the current bitstream can be decoded, the GPCC decoder
10 can gracefully terminate the decoding process. Therefore, forward compatibility is also highly desirable for designing an earlier version standard. In contrast, a GPCC v1 decoder without such ability may not parse even the high level syntax of GPCC v2bitstreams. As a result, all tools, profile and tier information will be unknown.

[00134] This disclosure proposes to add a syntax element (e.g., a one- or two-bit syntax element)
15 to support the above-mentioned abilities, i.e., both forward and back compatibilities between previous and later version standards.

[00135] In an embodiment of the invention, a coding specification version of the bitstream of point cloud may be identified according to a syntax element associated with the coding specification version in at least one of the headers.

20 [00136] In an embodiment of the invention, a syntax element specifying the length of a sequence extension data and a syntax element specifying the sequence extension data in at least one of the headers may be obtained when the syntax element associated with the coding specification version is equal to a preset value.

[00137] In an embodiment of the invention, values of the sequence extension data may be ignored
25 by using the decoder conforming to a first coding specification version according to the syntax element specifying the length of the sequence extension data and the syntax element specifying

the sequence extension data when the syntax element associated with the coding specification version of the bitstream of point cloud is equal to the preset value. The coding specification version of the bitstream of point cloud is a second coding specification version when the syntax element associated with the coding specification version is equal to the preset value. The coding specification with the second coding specification version may be specified later than the coding specification with the first coding specification version.

[00138] FIG. 13 is a flowchart of recognizing a coding specification version of the bitstream of point cloud according to an embodiment of the invention. Referring to FIG. 13, each step in the methods may be performed by a decoder using the G-PCC decoding process illustrated as FIG. 4. In step S1310, the decode may obtain a bitstream of point cloud. In step S1320, the decode may parse a plurality of headers coded in layer structure in the bitstream of point cloud. In step S1330, the decode may identify a coding specification version of the bitstream of point cloud according to a syntax element associated with the coding specification version in the sequence header.

[00139] In step S1340, the decode may determine whether the syntax element associated with the coding specification version is equal to a preset value. If step is determined as yes, in step S1350, the decode may obtain a syntax element specifying the length of a sequence extension data and a syntax element specifying the sequence extension data in at least one of the headers. It should be noted that, if the decode conforming to the first coding specification version and the bitstream of point cloud is coded according to the second coding specification version, the decoder may ignore the values of the sequence extension data according to the syntax element specifying the length of a sequence extension data. In step S1360, the decode may decode the bitstream of point cloud according to the headers.

[00140] FIG. 14A and 14B are schematic diagrams of a syntax table 300a of a sequence header according to an embodiment of the invention. Referring to FIG. 14A and 14B, a syntax element (e.g., **sequence_extension_enable**) associated with the coding specification version is included in the sequence header. In one exemplary case, **sequence_extension_enable** equal to 0x00

specifies that **sequence_extension_length** and **sequence_extension_data_byte** syntax elements are not present in the sequence header syntax structure, and **sequence_extension_enable** equal to 0x10 specifies that **sequence_extension_length** and **sequence_extension_data_byte** may be present in the sequence header syntax structure. Here, **sequence_extension_enable** shall be equal to 0x00 in bitstreams conforming to the first coding specification version of the current specification. However, **sequence_extension_enable** equal to 0x10 may be configured and specified due to the second coding specification version of the current specification, and the decoder conforming to the first coding specification version of the current specification shall allow the value of **sequence_extension_enable** equal to 0x10 to appear in the syntax.

[00141] In one exemplary case, **sequence_extension_length** specifies the length of the sequence extension data in bytes, not including the bits used for signalling **sequence_extension_length** itself. When **sequence_extension_length** is not present, the value of **sequence_extension_length** is inferred to be equal to 0. Although **sequence_extension_length** is not present in bitstreams conforming to the first coding specification version, **sequence_extension_length** could be used and specified in a specification with the second coding specification version. The decoders conforming to the first coding specification version shall allow **sequence_extension_length** to be present and in the range of 0 to N, e.g. N is 256, inclusive.

[00142] In one exemplary case, in FIG. 14A, **sequence_extension_data_byte[i]** may have any value. The presence and value of **sequence_extension_data_byte[i]** do not affect the decoding process specified in the first coding specification version. The decoder conforming to the first coding specification version shall ignore the values of all the **sequence_extension_data_byte[i]** syntax elements. Its value does not affect the decoding process specified in the first coding specification version of the specification.

[00143] In one exemplary case, in FIG. 14B, **sequence_extension_data_bit** may have any value. The presence and value of **sequence_extension_data_bit** do not affect the decoding process specified in the first coding specification version of this Specification. The decoders conforming

to the first coding specification version shall ignore the values of all the **sequence_extension_data_bit** syntax elements. Its value does not affect the decoding process specified in the first coding specification version of the specification.

[00144] In one exemplary case, **more_data_in_sequence_header_extension()** is specified as follows:

[00145] – If (the current position in the **sequence_header()** syntax structure) – (the position immediately following **sequence_extension_length**) is less than (**sequence_extension_length** * 8), the return value of **more_data_in_sequence_header_extension()** is equal to TRUE.

[00146] – Otherwise, the return value of **more_data_in_sequence_header_extension()** is equal to FALSE.

[00147] FIG. 15A and 15B are schematic diagrams of a syntax table 400a, 400b of a geometry header according to an embodiment of the invention. Referring to FIG. 15A and 15B, a syntax element (e.g., **geometry_extension_enable**) associated with the coding specification version is included in the sequence header. In one exemplary case, **geometry_extension_enable** equal to 0x00 specifies that **geometry_extension_length** and **geometry_extension_data_byte** syntax elements are not present in the sequence header syntax structure, and **geometry_extension_enable** equal to 0x10 specifies that **geometry_extension_length** and **geometry_extension_data_byte** may be present in the sequence header syntax structure.

[00148] The definition of **geometry_extension_enable**, **geometry_extension_length**, **geometry_extension_data_byte[i]** and **geometry_extension_data_bit** is similar with the definition of **sequence_extension_enable**, **sequence_extension_length**, **sequence_extension_data_byte[i]** and **sequence_extension_data_bit**, and thus those skilled in the art can infer the usage and definitions of these grammatical elements with reference to the foregoing description.

[00149] FIG. 16A and 16B are schematic diagrams of a syntax table 900a, 900b of an attribute header according to an embodiment of the invention. Referring to FIG. 16A and 16B, a syntax

element (e.g., **attribute_extension_enable**) associated with the coding specification version is included in the sequence header. In one exemplary case, **attribut_extension_enable** equal to 0x00 specifies that **attribute_extension_length** and **attribute_extension_data_byte** syntax elements are not present in the sequence header syntax structure, and **attribute_extension_enable** equal to 0x10 specifies that **attribute_extension_length** and **attribute_extension_data_byte** may be present in the sequence header syntax structure.

[00150] The definition of **attribute_extension_enable**, **attribute_extension_length**, **attribute_extension_data_byte[i]** and **attribute_extension_data_bit** is similar with the definition of **sequence_extension_enable**, **sequence_extension_length**, **sequence_extension_data_byte[i]** and **sequence_extension_data_bit**, and thus those skilled in the art can infer the usage and definitions of these grammatical elements with reference to the foregoing description.

[00151] FIG. 17A and 17B are schematic diagrams of a syntax table 1700a, 1700b of a geometry slice header according to an embodiment of the invention. Referring to FIG. 17A and 17B, a syntax element (e.g., **geometry_slice_extension_enable**) associated with the coding specification version is included in the sequence header. In one exemplary case, **geometry_slice_extension_enable** equal to 0x00 specifies that **geometry_slice_extension_length** and **geometry_slice_extension_data_byte** syntax elements are not present in the sequence header syntax structure, and **geometry_slice_extension_enable** equal to 0x10 specifies that **geometry_slice_extension_length** and **geometry_slice_extension_data_byte** may be present in the sequence header syntax structure.

[00152] The definition of **geometry_slice_extension_enable**, **geometry_slice_extension_length**, **geometry_slice_extension_data_byte[i]** and **geometry_slice_extension_data_bit** is similar with the definition of **sequence_extension_enable**, **sequence_extension_length**, **sequence_extension_data_byte[i]** and **sequence_extension_data_bit**, and thus those skilled in the art can infer the usage and

definitions of these grammatical elements with reference to the foregoing description.

[00153] FIG. 18A and 18B are schematic diagrams of a syntax table 1800a, 1800b of an attribute slice header according to an embodiment of the invention. Referring to FIG. 18A and 18B, a syntax element (e.g., **attribute_slice_extension_enable**) associated with the coding specification version is included in the sequence header. In one exemplary case, **attribute_slice_extension_enable** equal to 0x00 specifies that **attribute_slice_extension_length** and **attribute_slice_extension_data_byte** syntax elements are not present in the sequence header syntax structure, and **attribute_slice_extension_enable** equal to 0x10 specifies that **attribute_slice_extension_length** and **attribute_slice_extension_data_byte** may be present in the sequence header syntax structure.

[00154] The definition of **attribute_slice_extension_enable**, **attribute_slice_extension_length**, **attribute_slice_extension_data_byte[i]** and **attribute_slice_extension_data_bit** is similar with the definition of **sequence_extension_enable**, **sequence_extension_length**, **sequence_extension_data_byte[i]** and **sequence_extension_data_bit**, and thus those skilled in the art can infer the usage and definitions of these grammatical elements with reference to the foregoing description.

[00155] FIG. 19 is a schematic diagram of a syntax table 1900 of a sequence header according to an embodiment of the invention. Referring to FIG. 19, a syntax element (e.g., **profile_id**) associated with the coding specification version is included in the sequence header. In one exemplary case, **profile_id** equal to 0xXX specifies that **sequence_extension_length** and **sequence_extension_data_byte[i]** syntax elements are not present in the sequence header syntax structure. It should be noted that, the syntax table in FIG. 14A to 18B may be modified in a similar way by using the syntax element (e.g., **profile_id**) associated with the coding specification version.

[00156] This disclosure proposes to use a predefined extension and user data structures to support both forward and back compatibilities between previous and later version standards. In an

embodiment of the invention, the high level syntax or the new defined function of the later version standards may be presented in the extension data or the user data.

[00157] In an embodiment of the invention, a extension data and a user data are disposed after each of the headers, and the extension data and the user data are identified from the bitstream of point cloud according to a prefix bitstring and a predefined code. For example, in the syntax table 100 of FIG. 10A, a function for decoding the user data and the extension data (e.g., extension_and_user_data(0)) may be disposed after each of the sequence header, the geometry header, the attribute header, and the frame header.

[00158] In an embodiment of the invention, the prefix bitstring includes 24-bit binary code, and the predefined code includes an 8-bit binary code. For example, the prefix bitstring may be a 24-bit binary code "0000 0000 0000 0000 0000 0001".

[00159] In an embodiment of the invention, the predefined code includes an extension start code and a user data start code. For example, the coded value of the extension start code and the user data start code may be defined in Table 1.

[00160] For example, FIG. 20 is a schematic diagram of an extension and user data structure 2000 according to an embodiment of the invention. If the next 32 bits match to the prefix bitstring concatenated with the coded value of the extension start code or match to the prefix bitstring concatenated with the coded value of the user data start code, the decoder may obtain the extension data and the user data according to the coded value of the extension start code and the coded value of the user data start code. For example, FIG. 21A and FIG. 21B are schematic diagrams of an extension data structure 2100a, 2100b according to an embodiment of the invention. Besides, FIG. 22 is a schematic diagram of a user data structure 2200 according to an embodiment of the invention. Comparing FIG. 21A with FIG. 21B, the extension functions in the future coding version may be included in the extension data of FIG. 21A, and no any defined extension functions is included in the extension data of FIG. 21A since there is no any defined extension yet in the current coding version.

[00161] FIG. 23 is a flowchart of a point cloud encoding method applied to an encoder according to an embodiment of the invention. Referring to FIG. 23, each step in the methods may be performed by a encoder using the G-PCC encoding process illustrated as FIG. 1. In step S2310, the encoder may obtain point cloud data. In step S2320, the encoder may encode the point cloud data to generate a bitstream of point cloud comprising a plurality of headers coded in layer structure. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header. The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header. That is, the headers are disposed in a specific order within the bitstream. Each header may include information (e.g., parameters, identifiers, or flags) for decoding the bitstream of point cloud.

[00162] In an embodiment of the invention, the headers in the bitstream of point cloud may include an attribute slice header and a geometry slice header. The attribute slice header and the geometry slice header for each slice of a PCC frame may be inserted into the bitstream of point cloud by the encode when a syntax flag included in the frame header is equal to 1. The attribute slice header and the geometry slice header for only one slice of the PCC frame may be inserted into the bitstream of point cloud by the encode when a slice index is equal to 0 and the syntax flag included in the frame header is not equal to 1.

[00163] In an embodiment of the invention, a prefix bit string and a predefined code are inserted into the bitstream of point cloud by the encode to indicate the headers in the bitstream of point cloud, wherein the last bit of the prefix bitstring is concatenated with the first bit of the predefined code.

[00164] In an embodiment of the invention, a syntax element is inserted into at least one of the headers in the bitstream of point cloud by the encode according to a coding specification version of the bitstream of point cloud.

[00165] In an embodiment of the invention, a syntax element specifying the length of a sequence extension data and a syntax element specifying the sequence extension data are inserted into at least one of the headers when the syntax element associated with the coding specification version is equal to a preset value.

5 [00166] The decoder of the invention includes a storage device and a processor. The processor is electrically connected to the storage device, and is configured to: obtain a bitstream of point cloud; parse a plurality of headers coded in layer structure in the bitstream of point cloud; and decode the bitstream of point cloud according to the headers. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header.

10 The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

[00167] The encoder of the invention includes a storage device and a processor. The processor is electrically connected to the storage device, and is configured to: obtain point cloud data; and

15 encode the point cloud data to generate a bitstream of point cloud comprising a plurality of headers coded in layer structure. The headers in the bitstream of point cloud includes a sequence header, a geometry header, an attribute header, and a frame header. The sequence header is disposed outside the geometry header, the attribute header, and the frame header. The geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the

20 geometry header.

[00168] **[Aspect 4: Throughput related solution]**

[00169] This disclosure proposes that an individual counter is used to control how many bins are coded as context-coded bins for coding the value of zero-run length, the values of reflectance, and color residual, respectively. Once one context-coded bin is generated, the corresponding counter

25 increases by one. If the corresponding counter reaches a predefined value, all the remaining bins for coding the value of zero-run length, the values of reflectance, and color residual will be coded

with the bypass approach, respectively.

[00170] As an example, suppose that the value of zero-run length is 62, and a predefined value of context-coded bins for coding the value of zero-run length is set as 10. Before the coding, the counter is set to 0. The first bin will be coded to indicate the value of zero-run length is not zero and the counter increases by one; the second bin will be coded to indicate the value of zero-run length is not one and the counter increases by one; the third bin will be coded to indicate the value of zero-run length is not two and the counter increases by one; the fourth bin will be coded to indicate the parity of 62 and the counter increases by one. After these four bins are coded with the context-based approach, the remainder will be $(62-2)/2 = 30$, and 30 will be binarized with a 2nd-order EG code as 00001000010 with 11 bins. Because there are six more context-coded bins left in the budget, the first six bins 000010 will be coded with the context-based approach and the remaining five bins 00010 will be coded with the bypass approach. Alternatively, the first five 00001 bins will be coded with the bypass approach, and the rest six bins 000010 will be coded with the context-based approach. Similarly, predefined counter for reflectance and color residual will be used to control the throughput for reflectance and color residual coding, respectively.

[00171] Beyond the individual counter for zero-run length, reflectance and color residual coding, a global counter or several global counters are proposed to count total context bins for all bins, including geometry coding, zero-run length, reflectance and color residual coding etc. so that the total context bins are less than a predefined value. The global counter could work at frame level, it also can work at subset of coded point cloud level. For example, the total context bins for K coded point cloud could be less than the predefined value where number K and the predefined values could be related to profile definition.

[00172] FIG. 24 is a schematic block diagram of an codec device provided by an embodiment of the invention. Referring to FIG. 24, the codec device 1000 may be the decoder or the encoder described in an embodiment of the invention. The codec device 1000 may comprise a transceiver 1001, a storage device 1002 and a processor 1003.

[00173] The storage device 1002 may stores executable instructions or the computer program. The processor 1003 may execute the executable instructions or the computer program to implement the methods in an embodiment of the invention. For example, the processor 1003 may be used to perform the steps of the above method in FIG. 8 to FIG. 11 according to the instructions or the computer program stored in the storage device 1002.

[00174] In some embodiments of the invention, the processor 1003 may include (but is not limited to) a central processing unit (CPU), a graphic processing unit (GPU), general-purpose or special-purpose processors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs) or other programmable logic devices, discrete gate or transistor logic devices, discrete hardware components, etc. In one embodiment of the invention, the codec device 1000 may include more processors to implement the method in an embodiment of the invention.

[00175] In some embodiments of the invention, the storage device 1002 includes (but is not limited to) volatile memory and/or non-volatile memory. Among them, non-volatile memory can be Read-Only Memory (ROM), programmable ROM (PROM), erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM) or flash memory. Volatile memory can be Random Access Memory (RAM), which is used as an external cache. By illustrative but not limitless, many forms of RAM are available, such as Static RAM (SRAM), Dynamic RAM (DRAM), Synchronous DRAM (SDRAM), Double Data Rate SDRAM (DDR SDRAM), Enhanced SDRAM (ESDRAM), synch link DRAM (SLDRAM), and Direct Rambus RAM (DR RAM).

[00176] In some embodiments of the invention, the computer program may be divided into one or more units, the one or more units are stored in the storage device 1002, and executed by the processor 1003 to complete the method provided in the invention. The unit or units may be a series of computer program instruction segments capable of completing a particular function.

[00177] The transceiver 1001 may be connected to the processor 1003 or the storage device 1002.

The processor 1003 may control the transceiver 1003 to communicate with other devices, specifically, may send information or data to other devices, or receive information or data sent by other devices. For example, the transceiver 1003 is configured to receive or transmit the bitstream of point cloud. The transceiver 1001 may include a transmitter and receiver. The
5 transceiver 1001 may further include antennas, the number of antennas may be one or more.

[00178] It should be understood that the various components in the codec device 1000 are connected by a bus system, wherein the bus system includes a power bus, a control bus and a status signal bus in addition to the data bus.

[00179] The invention also provides a computer storage medium on which a computer program
10 is stored, and the computer program is executed by the computer such that the computer can execute the method embodiment of the above method. In other words, embodiments of the invention also provide a computer program product comprising instructions, the instruction is executed by the computer when the computer executes the method of executing the above method embodiment.

[00180] When implemented using software, it may be implemented in whole or in part in the
15 form of a computer program product. The computer program product includes one or more computer instructions. When loading and executing the computer program instructions on the computer, the process or function in accordance with the invention is produced in whole or in part. The computer may be a general-purpose computer, a special purpose computer, a computer
20 network, or other programmable device. The computer instruction may be stored in a computer-readable storage medium, or transmitted from one computer-readable storage medium to another computer-readable storage medium, for example, the computer instruction may be transmitted from a website site, computer, server or data center to another website site (e.g., coaxial cable, optical fiber, digital subscriber line (DSL)) or wireless (e.g., infrared, wireless, microwave, etc.)
25 to another website site, computer, Server or data center for transmission. The computer-readable storage medium may be any available media that the computer can access, or a data storage device

such as a server, data center, etc., that contains one or more available media integration. The available media may be magnetic media (e.g., floppy disks, hard disks, magnetic tapes), optical media (e.g., digital video discs (DVDs)), or semiconductor media (e.g., solid state disks (SSDs)) and the like.

5 **[00181]** the G-PCC system applying the above-mentioned improved decoding method and the above-mentioned improved encoding method may adapt to a wide range of PCC inputs and can be used in many applications.

[00182] It will be apparent to those skilled in the art that various modifications and variations can be made to the disclosed embodiments without departing from the scope or spirit of the disclosure.

10 In view of the foregoing, it is intended that the disclosure covers modifications and variations provided that they fall in the scope of the following claims and their equivalents.

Reference Signs List

[00183]

15 1000 : codec device

 1003 : Processor

 1002 : Storage device

 1001 : Transceiver

 100, 200, 300a, 300b, 400a, 400b, 500a, 500b, 600, 700, 900a, 900b, 1700a, 1700b, 1800a,

20 1800b, 1900 : syntax table

 B1~B8 : Cube

 2000: extension and user data structure

 2100a, 2100b: extension data structure

 2200: user data structure

25 S102、S104、S106、S108、S110、S112、S114、S116、S118、S120、S122、S124、

S126、S402、S404、S406、S408、S410、S412、S414、S416、S418、S420、S422、S810、
S820、S830、S821、S822、S823、S825、S826、S1310、S1320、S1330、S1340、S1350、
S1356、S2310、S2320：Step

WHAT IS CLAIMED IS:

1. A point cloud decoding method, comprising:

obtaining a bitstream of point cloud; and

parsing a plurality of headers and point cloud data coded in layer structure in the bitstream

5 of point cloud,

wherein the headers in the bitstream of point cloud comprises a sequence header, a geometry header, an attribute header, and a frame header,

wherein the sequence header is disposed outside the geometry header, the attribute header, and the frame header,

10 the geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

2. The point cloud decoding method according to claim 1, wherein the headers in the bitstream of point cloud further comprises an attribute slice header and a geometry slice header,

15 the geometry slice header is before the attribute slice header, and the step of parsing the headers coded and the point cloud data in layer structure in the bitstream of point cloud comprises:

when a syntax flag included in the frame header is equal to 1, parsing the attribute slice header and the geometry slice header for each slice of a PCC frame; and

20 when a slice index is equal to 0 and the syntax flag included in the frame header is not equal to 1, parsing the attribute slice header and the geometry slice header for only one slice of the PCC frame.

3. The point cloud decoding method according to claim 1, wherein the step of parsing the headers coded and the point cloud data in layer structure in the bitstream of point cloud comprises:

25 identifying the headers from the bitstream of point cloud according to a prefix bitstring and a predefined code, wherein the last bit of the prefix bitstring is concatenated with the first bit of

the predefined code.

4. The point cloud decoding method according to claim 3, wherein the prefix bitstring comprises a 24-bit binary code, and the predefined code comprises an 8-bit binary code.

5

5. The point cloud decoding method according to claim 3, wherein the predefined code comprises at least one of a slice start code, a slice end code, a pcc sequence start code, a pcc sequence end code, an attribute start code, a geometry start code, a frame start code, and a pcc edit code.

10

6. The point cloud decoding method according to claim 1, wherein the frame header comprises a first syntax element specifying an order of a PCC frame associated with a plurality of point cloud frames, the first syntax element is configured to uniquely identifies the associated PCC frame among all the point cloud frames, and the first syntax element is further configured to indicate a position of the associated PCC frame in output order relative to output order positions of other point cloud frames that are to be output from a decoded point cloud buffer when the associated PCC frame is to be output from the decoded point cloud buffer.

15

7. The point cloud decoding method according to claim 1, wherein the frame header comprises a second syntax element specifying a frame type of a PCC frame.

20

8. The point cloud decoding method according to claim 1, wherein the frame header comprises a third syntax element specifying whether there is more than one slice in a PCC frame.

25

9. The point cloud decoding method according to claim 1, wherein the frame header comprises a third syntax element specifying whether there is more than one slice in a PCC frame,

and a fourth syntax element specifying the number of slices in the PCC frame and a fifth syntax element specifying whether multiple geometry slice headers and multiple attribute slice headers are used for the slices in the frame header when the third syntax element is equal to a preset value.

5 10. The point cloud decoding method according to claim 1, wherein a sixth syntax element specifying whether a current intra-coded PCC frame is a random access point and a seventh syntax element specifying whether a delta QP is enabled for a PCC frame are present in the frame header when a second syntax element specifying a frame type of the PCC frame is equal to a preset value.

10 11. The point cloud decoding method according to claim 1, wherein the method further comprises:

identifying a coding specification version of the bitstream of point cloud according to a syntax element associated with the coding specification version in at least one of the headers.

15 12. The point cloud decoding method according to claim 11, wherein the method further comprises:

obtaining a syntax element specifying the length of a sequence extension data and a syntax element specifying the sequence extension data in at least one of the headers when the syntax element associated with the coding specification version is equal to a preset value.

20 13. The point cloud decoding method according to claim 12, wherein the method further comprises:

ignoring values of the sequence extension data by using a decoder conforming to a first coding specification version according to the syntax element specifying the length of the sequence extension data and the syntax element specifying the sequence extension data when the syntax element associated with the coding specification version of the bitstream of point cloud is equal

25

to the preset value,

wherein the coding specification version of the bitstream of point cloud is a second coding specification version when the syntax element associated with the coding specification version is equal to the preset value.

5

14. The point cloud decoding method according to claim 1, wherein a extension data and a user data are disposed after each of the headers, and the extension data and the user data are identified from the bitstream of point cloud according to a prefix bitstring and a predefined code.

10 15. The point cloud decoding method according to claim 14, wherein the prefix bitstring comprises a 24-bit binary code, and the predefined code comprises an 8-bit binary code.

16. The point cloud decoding method according to claim 14, wherein the predefined code comprises an extension start code and a user data start code.

15

17. A point cloud encoding method, comprising:

obtaining point cloud data; and

encoding the point cloud data to generate a bitstream of point cloud comprising a plurality of headers coded in layer structure,

20 wherein the headers in the bitstream of point cloud comprises a sequence header, a geometry header, an attribute header, and a frame header,

wherein the sequence header is disposed outside the geometry header, the attribute header, and the frame header,

25 the geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

18. The point cloud encoding method according to claim 17, wherein the headers in the bitstream of point cloud further comprises an attribute slice header and a geometry slice header, the geometry slice header is before the attribute slice header, and the step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer structure
5 further comprises:

inserting the attribute slice header and the geometry slice header for each slice of a PCC frame into the bitstream of point cloud when a syntax flag included in the frame header is equal to 1; and

10 inserting the attribute slice header and the geometry slice header for only one slice of the PCC frame into the bitstream of point cloud when a slice index is equal to 0 and the syntax flag included in the frame header is not equal to 1.

19. The point cloud encoding method according to claim 17, wherein the step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer
15 structure further comprises:

inserting a prefix bitstring and a predefined code into the bitstream of point cloud to indicate the headers in the bitstream of point cloud, wherein the last bit of the prefix bitstring is concatenated with the first bit of the predefined code.

20 20. The point cloud encoding method according to claim 19, wherein the prefix bitstring comprises 24-bit binary code, and the predefined code comprises 8-bit binary code.

21. The point cloud encoding method according to claim 19, wherein the predefined code comprises at least one of a slice start code, a slice end code, a pcc sequence start code, a pcc
25 sequence end code, an attribute start code, a geometry start code, a frame start code, and a pcc edit code.

22. The point cloud encoding method according to claim 17, wherein the frame header comprises a first syntax element specifying an order of a PCC frame associated with a plurality of point cloud frames, the first syntax element is configured to uniquely identify the associated PCC frame among all the point cloud frames, and the first syntax element is further configured to indicate a position of the associated PCC frame in output order relative to output order positions of other point cloud frames that are to be output from a decoded point cloud buffer when the associated PCC frame is to be output from the decoded point cloud buffer.

23. The point cloud encoding method according to claim 17, wherein the frame header comprises a second syntax element specifying a frame type of a PCC frame.

24. The point cloud encoding method according to claim 17, wherein the frame header comprises a third syntax element specifying whether there is more than one slice in a PCC frame.

25. The point cloud encoding method according to claim 17, wherein the frame header comprises a third syntax element specifying whether there is more than one slice in a PCC frame, and a fourth syntax element specifying the number of slices in the PCC frame and a fifth syntax element specifying whether multiple geometry slice headers and multiple attribute slice headers are used for the slices are present in the frame header when the third syntax element is equal to a preset value.

26. The point cloud encoding method according to claim 17, wherein a sixth syntax element specifying whether a current intra-coded PCC frame is a random access point and a seventh syntax element specifying whether a delta QP is enabled for a PCC frame are present in the frame header when a second syntax element specifying a frame type of the PCC frame is equal to a preset value.

27. The point cloud encoding method according to claim 17, wherein the step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer structure further comprises:

5 inserting a syntax element according to a coding specification version of the bitstream of point cloud into at least one of the headers in the bitstream of point cloud.

28. The point cloud encoding method according to claim 27, wherein the step of encoding the point cloud data to generate the bitstream of point cloud comprising the headers coded in layer structure further comprises:

10 inserting a syntax element specifying the length of a sequence extension data and a syntax element specifying the sequence extension data into at least one of the headers when the syntax element associated with the coding specification version is equal to a preset value.

15 29. The point cloud encoding method according to claim 17, wherein an extension data and a user data are disposed after each of the headers, and the extension data and the user data are inserted into the bitstream of point cloud according to a prefix bitstring and a predefined code.

20 30. The point cloud encoding method according to claim 29, wherein the prefix bitstring comprises 24-bit binary code, and the predefined code comprises 8-bit binary code.

31. The point cloud encoding method according to claim 29, wherein the predefined code comprises an extension start code and a user data start code.

25 32. A decoder, comprising:
a storage device; and

a processor, electrically connected to the storage device, and configured to:

obtain a bitstream of point cloud; and

parse a plurality of headers and point cloud data coded in layer structure in the bitstream of point cloud,

5 wherein the headers in the bitstream of point cloud comprises a sequence header, a geometry header, an attribute header, and a frame header,

wherein the sequence header is disposed outside the geometry header, the attribute header, and the frame header,

the geometry header and the attribute header are disposed outside the frame header, and the
10 attribute header is after the geometry header.

33. An encoder, comprising:

a storage device; and

a processor, electrically connected to the storage device, and configured to:

15 obtain point cloud data; and

encode the point cloud data to generate a bitstream of point cloud comprising a plurality of headers coded in layer structure,

wherein the headers in the bitstream of point cloud comprises a sequence header, a geometry header, an attribute header, and a frame header,

20 wherein the sequence header is disposed outside the geometry header, the attribute header, and the frame header,

the geometry header and the attribute header are disposed outside the frame header, and the attribute header is after the geometry header.

25

1/34

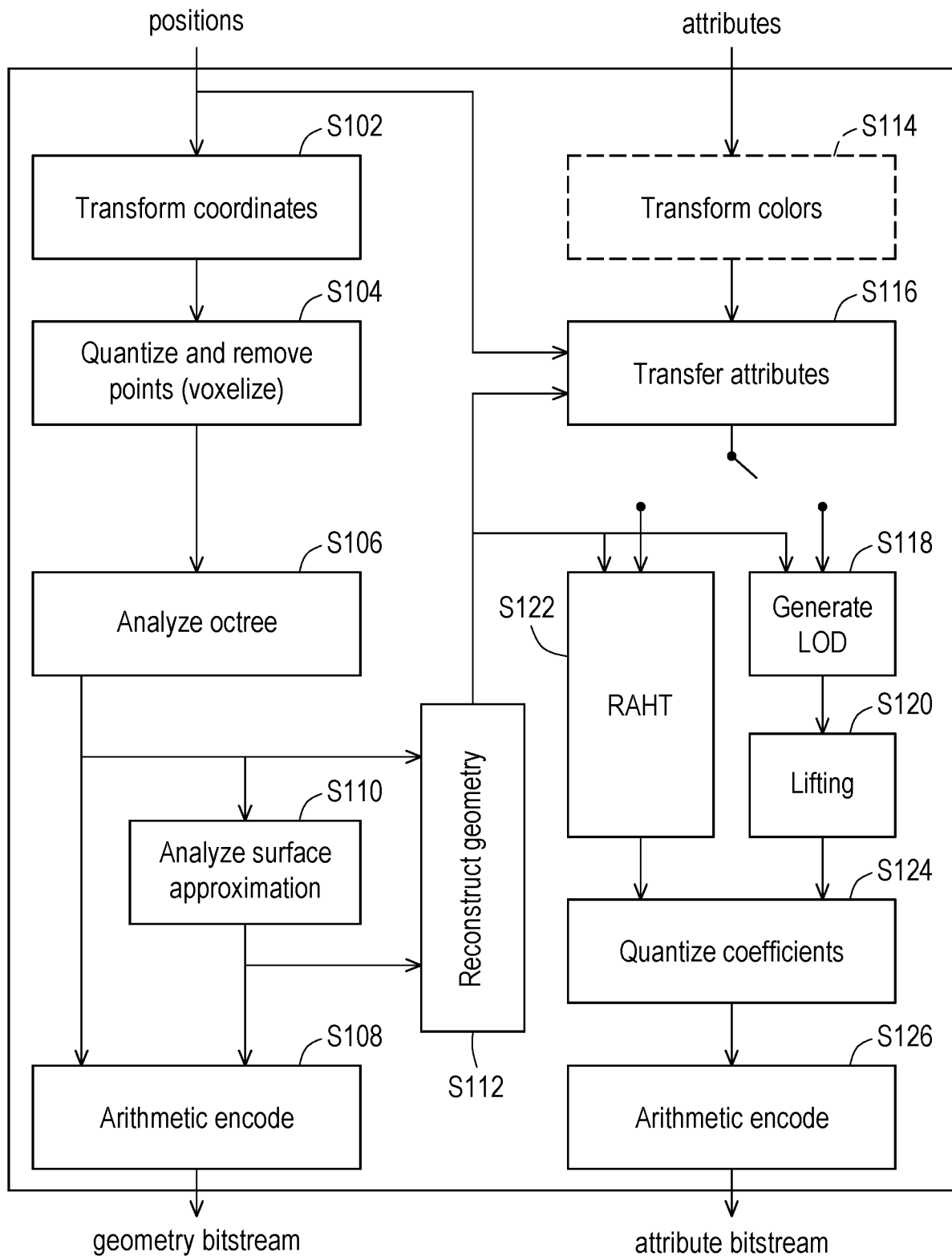


FIG. 1

2/34

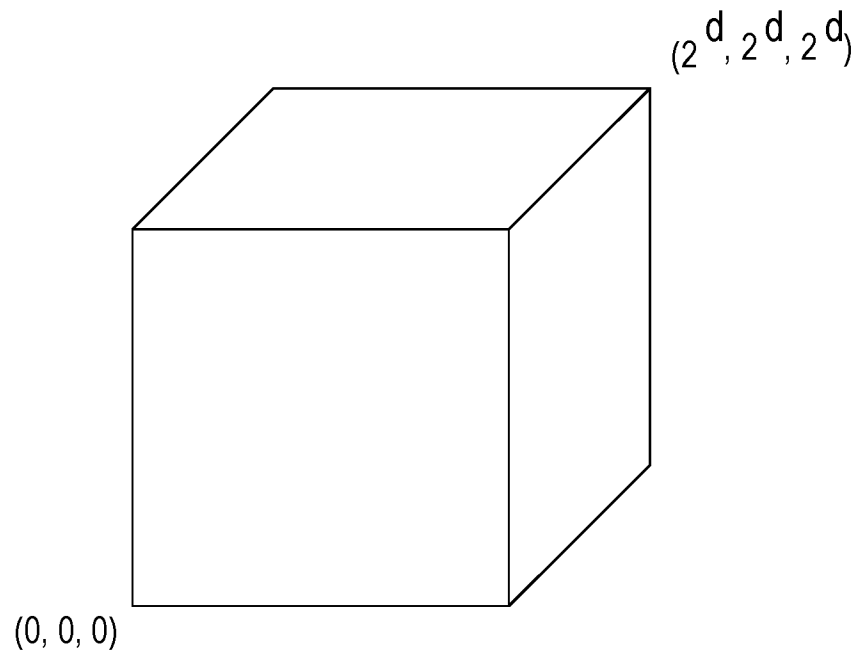


FIG. 2A

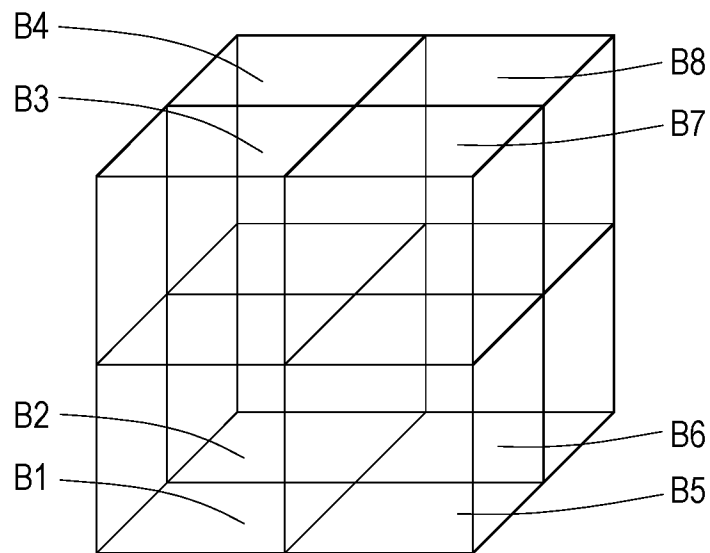


FIG. 2B

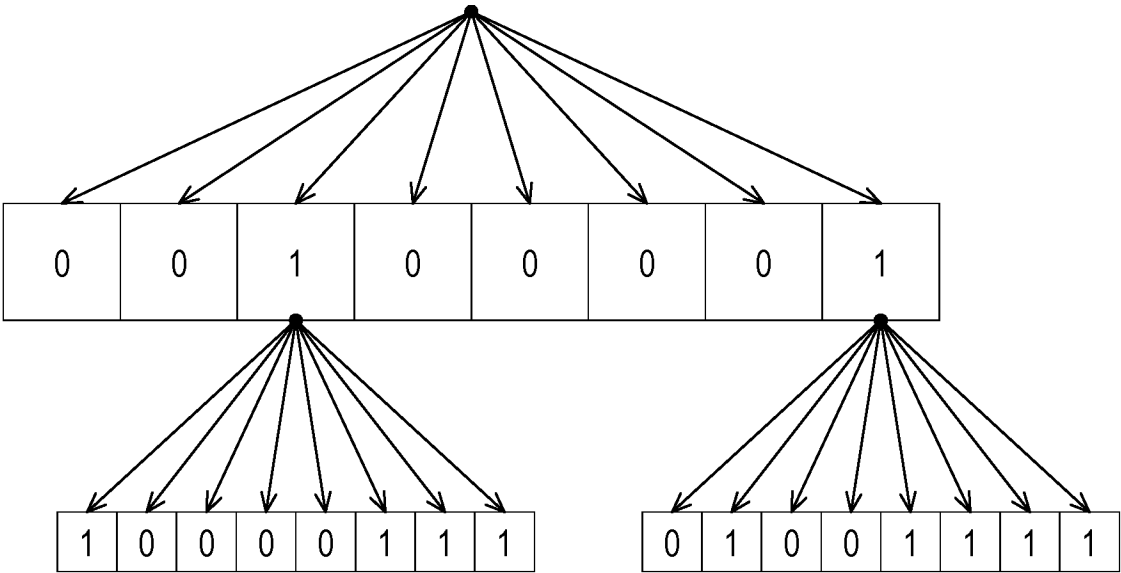


FIG. 2C

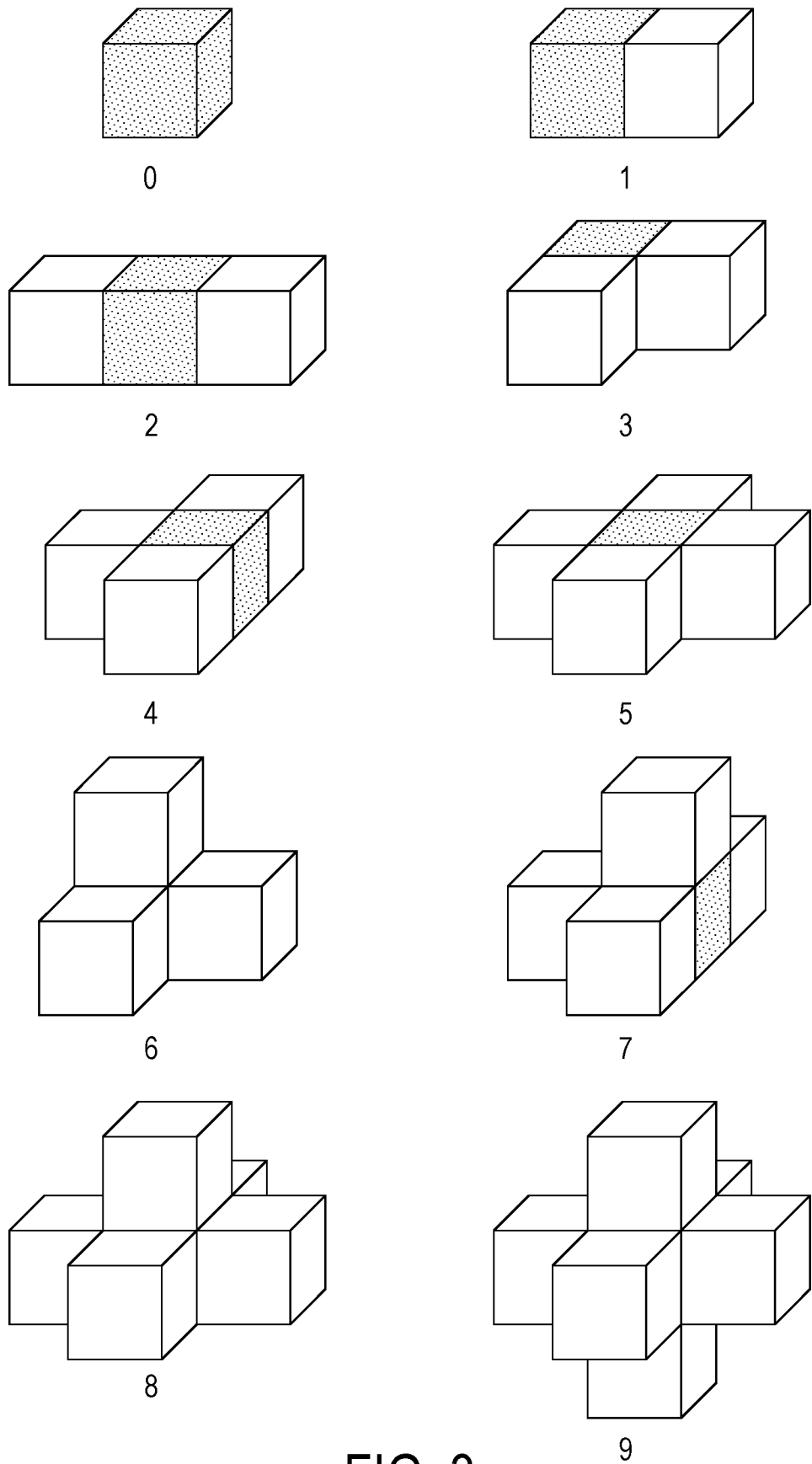


FIG. 3

5/34

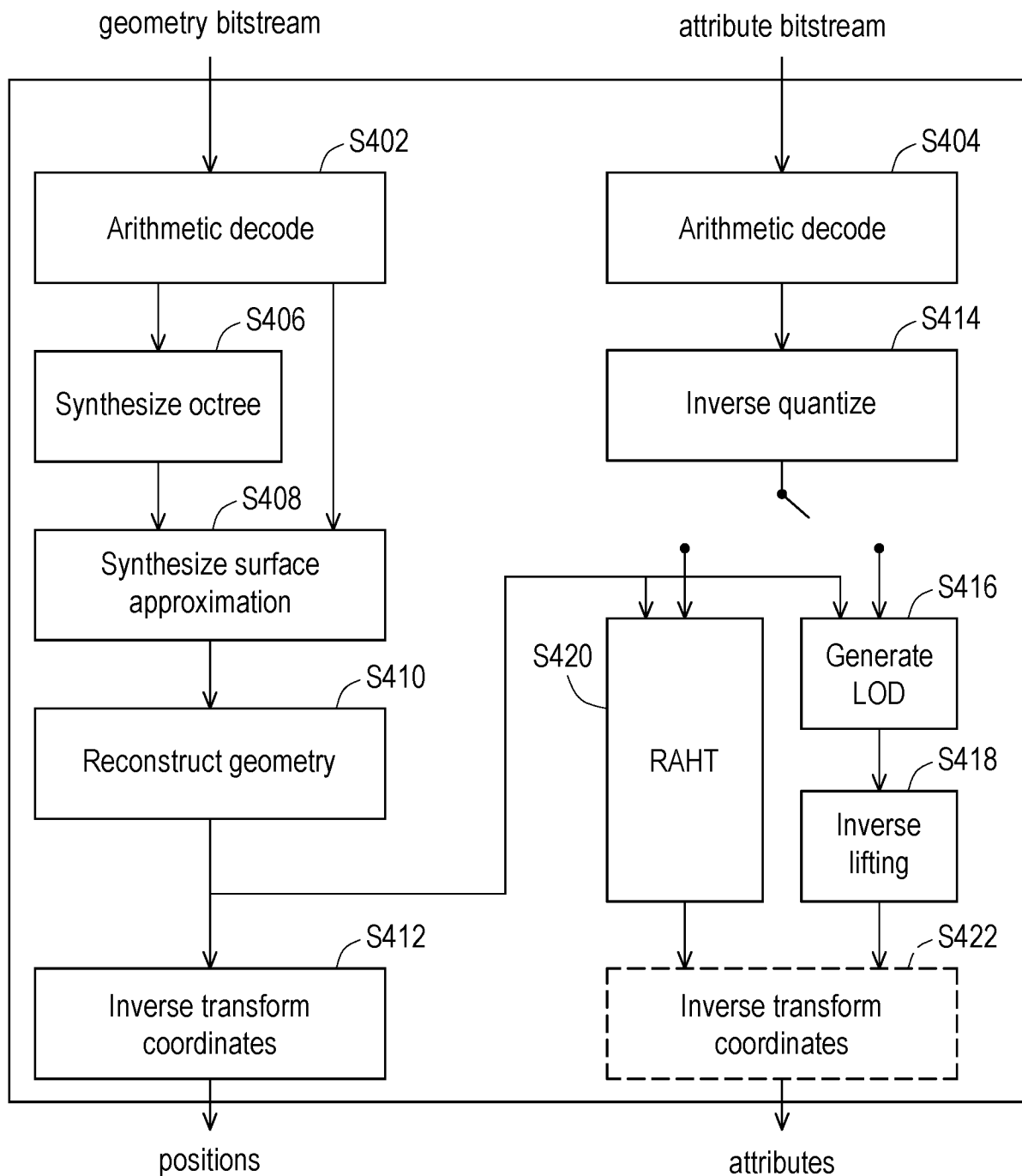


FIG. 4

500a

color_residual_coding() {
color_first_comp_zero
if (!color_first_comp_zero) {
color_component[0] = (2*coded_level_sign -1) * coded_level_coding (true) + 1
color_component[1] = coded_level_coding (false)
color_component[2] = coded_level_coding (false)
} else {
color_component[0] = 0
color_second_comp_zero
If (!color_second_comp_zero) {
color_component[1] = (2*coded_level_sign -1) * coded_level_coding (true) + 1
color_component[2] = coded_level_coding (false)
else {
color_component[1] = 0
color_component[2] = (2*coded_level_sign -1) * coded_level_coding (true) + 1
}
}
}

FIG. 5A

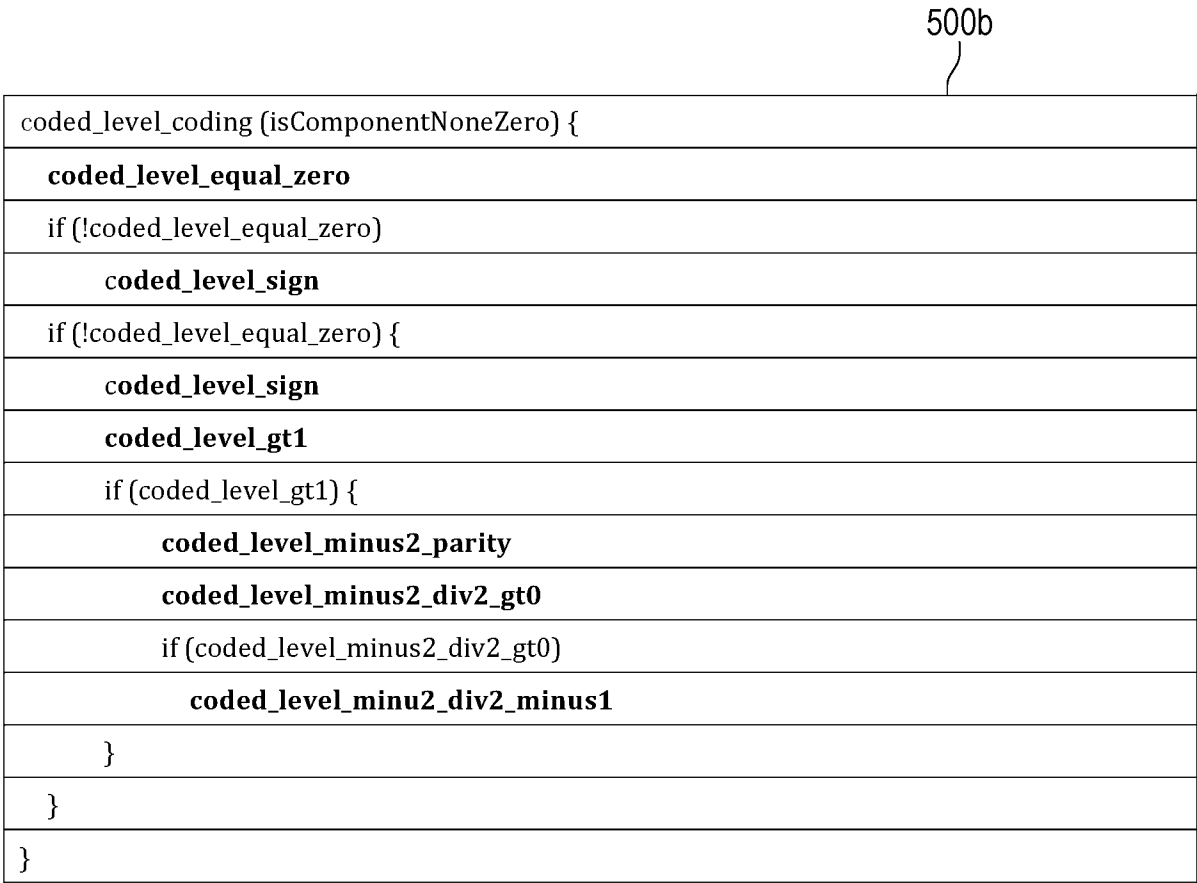


FIG. 5B

8/34

600

attribute_data_reflectance() {
zero_run_length_code(useGolomb)
for(i = 0; i < pointCloudNum; i++) {
if (residual_zero_run_length) {
-- residual_zero_run_length
} else {
if (!isDuplicatePoint) {
residual_sign
}
abs_level_minus1_parity
abs_level_minus1_div2_gt0
if (abs_level_minus1_div2_gt0) {
abs_level_minus1_div2_gt1
if (abs_level_minus1_div2_gt1) {
abs_level_minus1_div2_minus2
}
}
zero_run_length_code(useGolomb)
}
}
termination_bit_one /* equals to 1 */
}

FIG. 6

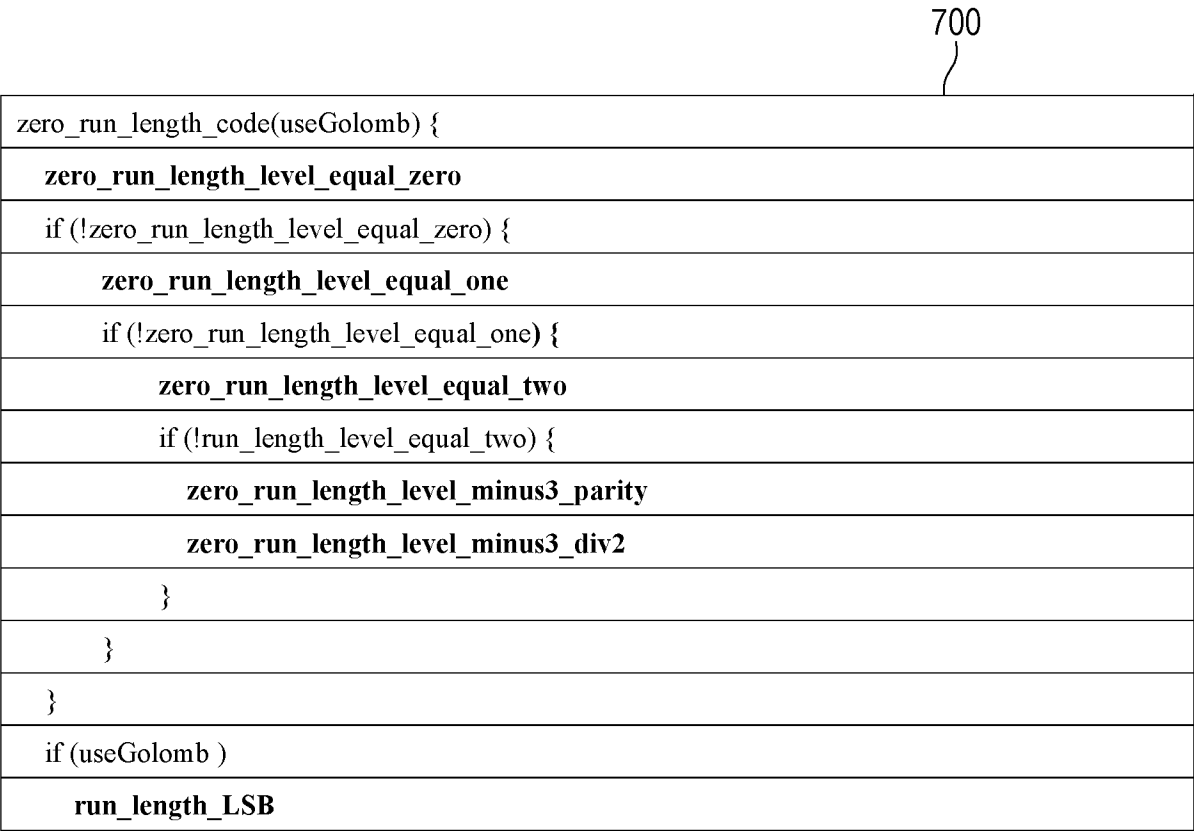


FIG. 7

10/34

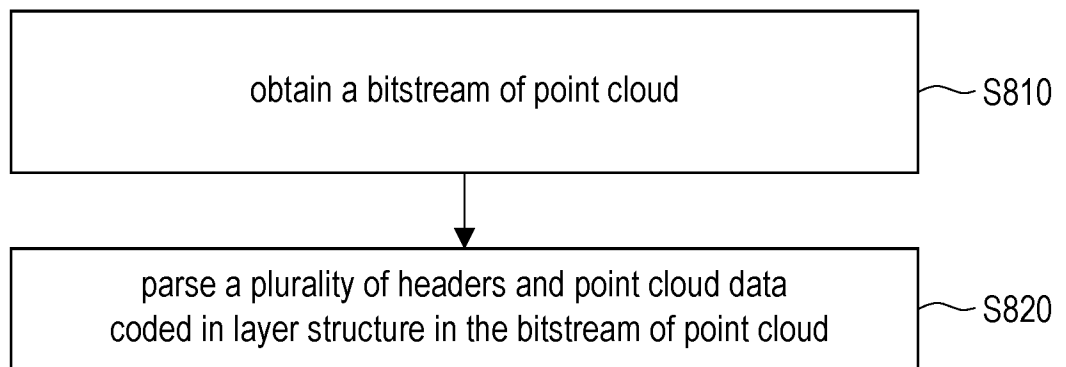


FIG. 8

11/34

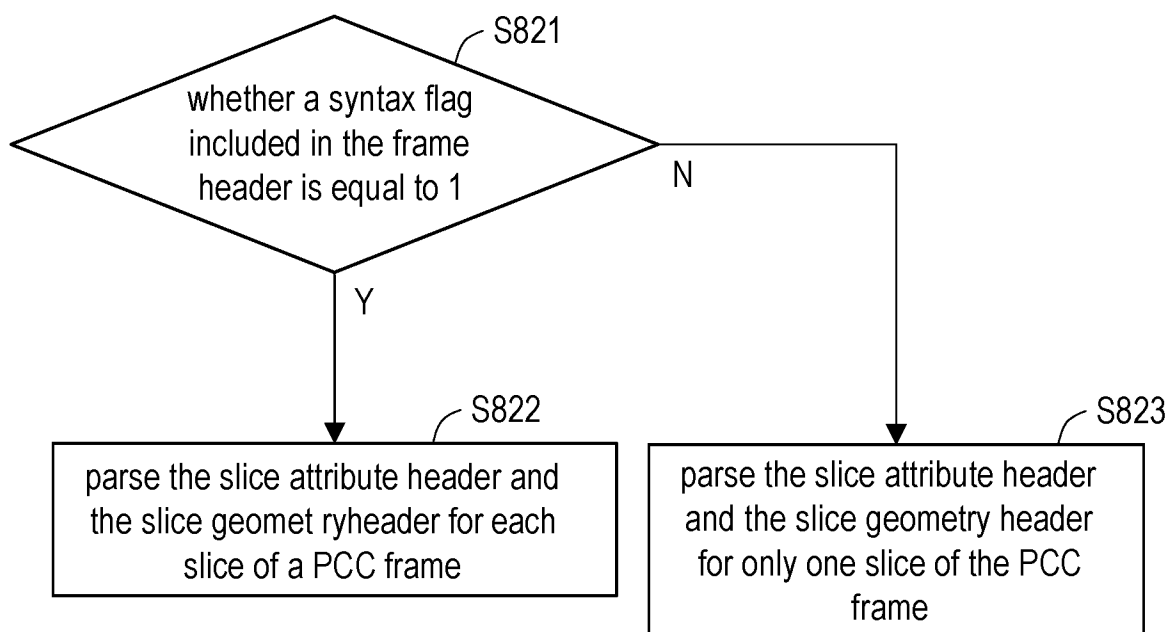


FIG. 9

100

PCC_sequence(){	Descriptor
do{	
sequence_header()	
geometry_header()	
if (attribute_present_flag) {	
attribute_header()	
}	
do{	
frame_header()	
do {	
geometry_slice()	
if (attribute_present_flag) {	
attribute_slice()	
}	
}while(next_bits(32) == slice_start_code)	
}while(next_bits(32) == frame_start_code)	
}while (next_bits(32) != pcc_sequence_end_code))	
}	

FIG. 10A

geometry_slice() {	Descriptor
geometry_slice_header()	
general_geometry_data_bitstream()	
}	

FIG. 10B

attribute_slice() {	Descriptor
attribute_slice_header()	
general_attribute_data_bitstream()	
}	

FIG. 10C

15/34

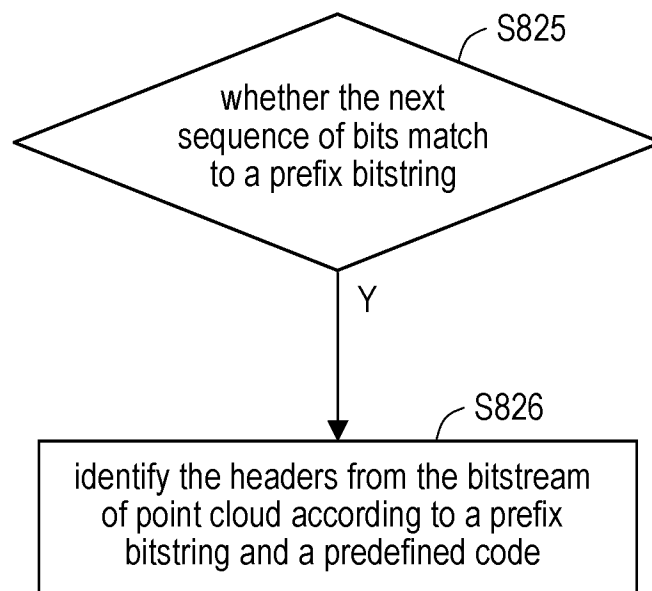


FIG. 11

200

frame_header() {	Descriptor
decode_order_index	u(8)
frame_type	u(2)
mul_slice_flag	u(1)
if(mul_slice_flag){	
num_slice_minus_two	ue(v)
mult_slice_header_enabled_flag	u(1)
}	
if(pic_type == 00)	
random_access_point_flag // random access point	u(1)
frame_qp_delta_enabled_flag // enable delta QP for current picture	u(1)
byte_alignment()	
}	

FIG. 12

17/34

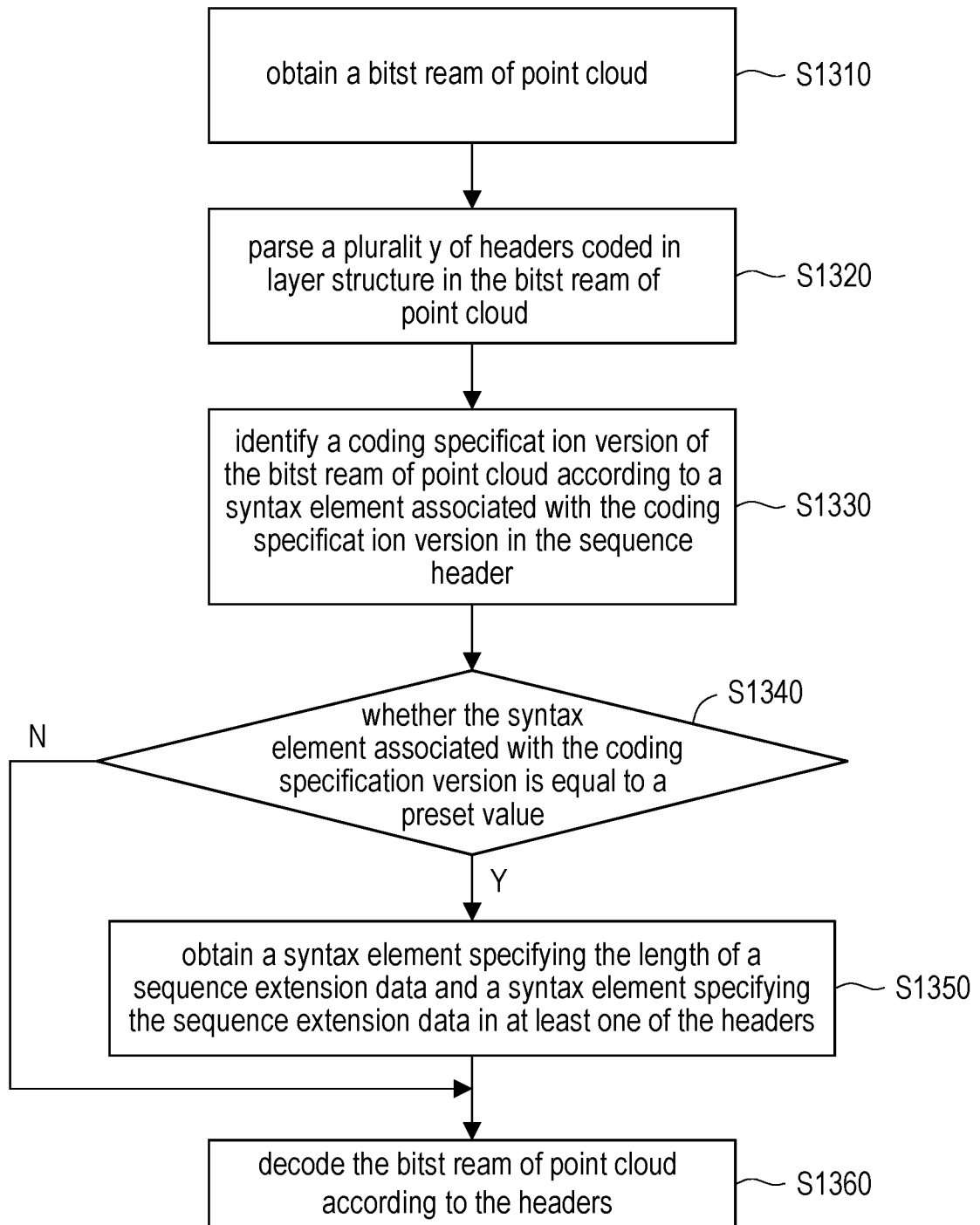


FIG. 13

300a

sequence_header() {	Descriptor
profile_id	u(8)
level_id	u(8)
.....	
attribute_present_flag	
if (attribute_present_flag) {	
attribute_adapt_pred	u(1)
colorQuantParam	ue(v)
reflQuantParam	ue(v)
maxNumAttributesMinus1	u(4)
}	
sequence_extension_enable	u(2)
if(sequence_extension_enable==0x10) {	
sequence_extension_length	ue(v)
for(i = 0; i < sequence_extension_length; i++)	
sequence_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

FIG. 14A

300b

sequence_header()	Descriptor
profile_id	u(8)
level_id	u(8)
.....	
attribute_present_flag	
if (attribute_present_flag) {	
attribute_adapt_pred	u(1)
colorQuantParam	ue(v)
reflQuantParam	ue(v)
maxNumAttributesMinus1	u(4)
}	
sequence_extension_enable	u(2)
if(sequence_extension_enable==0x10){	
sequence_extension_length	ue(v)
while (more_data_in_sequence_header_extension())	
sequence_extension_data_bit	u(1)
}	
byte_alignment()	
}	

FIG. 14B

400a

geometry_header() {	Descriptor
gps_lcu_node_size_log2_minus_one	ue(v)
gps_implicit_geom_partition_flag	u(1)
if(gps_implicit_geom_partition_flag) {	
gps_max_num_implicit_qtbt_before_ot	ue(v)
gps_min_size_implicit_qtbt	ue(v)
}	
gps_single_mode_flag	u(1)
gps_save_state_flag	u(1)
geometry_extension_enable	u(2)
if(geometry_extension_enable == 0x10){	
geometry_extension_length	ue(v)
for(i = 0; i < geometry_extension_length; i++)	
geometry_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

FIG. 15A

400b
}

geometry_header() {	Descriptor
gps_lcu_node_size_log2_minus_one	ue(v)
gps_implicit_geom_partition_flag	u(1)
if(gps_implicit_geom_partition_flag) {	
gps_max_num_implicit_qtbt_before_ot	ue(v)
gps_min_size_implicit_qtbt	ue(v)
}	
gps_single_mode_flag	u(1)
gps_save_state_flag	u(1)
geometry_extension_enable	u(2)
if(geometry_extension_enable == 0x10){	
geometry_extension_length	ue(v)
while (more_data_in_geometry_header_extension())	
geometry_extension_data_bit	u(1)
}	
byte_alignment()	
}	

FIG. 15B

900a

attribute_header()	Descriptor
.....	ue(v)
.....	
attribute_extension_enable	u(2)
if(attribute_extension_enable == 0x10){	
attribute_extension_length	ue(v)
for(i = 0; i < attribute_extension_length; i++)	
attribute_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

FIG. 16A

900b
/

attribute_header() {	Descriptor
....	ue(v)
.....	u(1)
attribute_extension_enable	u(2)
if(attribute_extension_enable == 0x10){	
attribute_extension_length	ue(v)
while (more_data_in_attribute_header_extension())	
attribute_extension_data_bit	u(1)
}	
byte_alignment()	
}	

FIG. 16B

1700a

geometry_slice_header() {	Descriptor
.....	ue(v)
.....	
geometry_slice_extension_enable	u(2)
if(geometry_slice_extension_enable == 0x10){	
geometry_slice_extension_length	ue(v)
for(i = 0; i < geometry_slice_extension_length; i++)	
geometry_slice_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

FIG. 17A

1700b
/

geometry_slice_header() {	Descriptor
....	ue(v)
.....	u(1)
geometry_slice_extension_enable	u(2)
if(geometry_slice_extension_enable == 0x10){	
geometry_slice_extension_length	ue(v)
while (more_data_in_geometry_slice_header_extension())	
geometry_slice_extension_data_bit	u(1)
}	
byte_alignment()	
}	

FIG. 17B

1800a

attribute_slice_header() {	Descriptor
.....	ue(v)
.....	
attribute_slice_extension_enable	u(2)
if(attribute_slice_extension_enable == 0x10){	
attribute_slice_extension_length	ue(v)
for(i = 0; i < attribute_slice_extension_length; i++)	
attribute_slice_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

FIG. 18A

1800b

attribute_slice_header() {	Descriptor
....	ue(v)
.....	u(1)
attribute_slice_extension_enable	u(2)
if(attribute_slice_extension_enable == 0x10){	
attribute_slice_extension_length	ue(v)
while (more_data_in_attribute_slice_header_extension())	
attribute_slice_extension_data_bit	u(1)
}	
byte_alignment()	
}	

FIG. 18B

1900
}

sequence_header() {	Descriptor
profile_id	u(8)
level_id	u(8)
.....	
attribute_present_flag	
if (attribute_present_flag) {	
attribute_adapt_pred	u(1)
colorQuantParam	ue(v)
reflQuantParam	ue(v)
maxNumAttributesMinus1	u(4)
}	
if(profile_id == 0xXX){	
sequence_extension_length	ue(v)
for(i = 0; i < sequence_extension_length; i++)	
sequence_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

FIG. 19

2000

extension_and_user_data(i) {	
while ((next_bits(32)== extension_start_code) (next_bits(32) == user_data_start_code)) {	
if (next_bits(32) == extension_start_code)	
extension_data(i)	
if (next_bits(32) == user_data_start_code)	
user_data()	
}	
}	

FIG. 20

30/34

2100a

extension_data(i) {	
while (next_bits(32) == extension_start_code) {	
extension_start_code	f(32)
if (i == 0) {	
if (next_bits(4) == '0010')	
sequence_display_extension()	
else if (next_bits(4) == '0011')	
temporal_scalability_extension()	
else if (next_bits(4) == '0100')	
copyright_extension()	
else if (next_bits(4) == '0110')	
cei_extension()	
else if (next_bits(4) == '1010')	
mastering_display_and_content_metadata_extension()	
else if (next_bits(4) == '1011')	
camera_parameters_extension()	
else if (next_bits(4) == '1101')	
cross_random_access_point_reference_extension()	
else	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
}	
else {	
if (next_bits(4) == '0100')	
copyright_extension()	
else if (next_bits(4) == '0101')	
hdr_dynamic_metadata_extension()	
else if (next_bits(4) == '0111')	
picture_display_extension()	
else if (next_bits(4) == '1011')	
camera_parameters_extension()	
else if (next_bits(4) == '1100')	
roi_parameters_extension()	
else {	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
}	
}	
}	

FIG. 21A

2100b

extension_data(i) {	
while (next_bits(32) == extension_start_code) {	
extension_start_code	f(32)
if (i == 0) {	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
else {	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
}	
}	
}	

FIG. 21B

2200

user_data() {	
user_data_start_code	f(32)
while (next_bits(24) != '0000 0000 0000 0000 0000 0001') {	
user_data	b(8)
}	
}	

FIG. 22

33/34

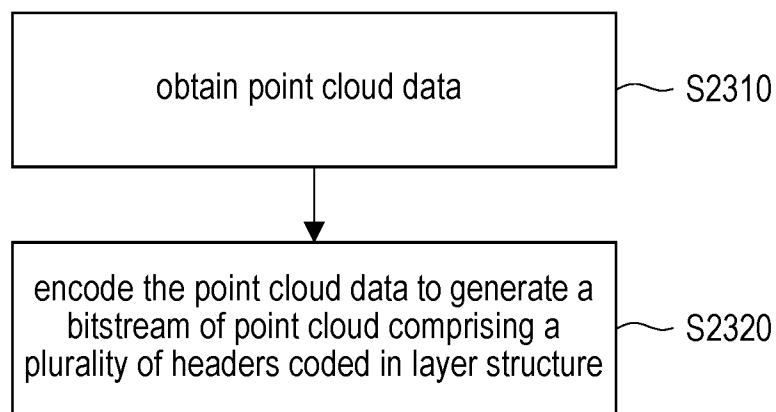


FIG. 23

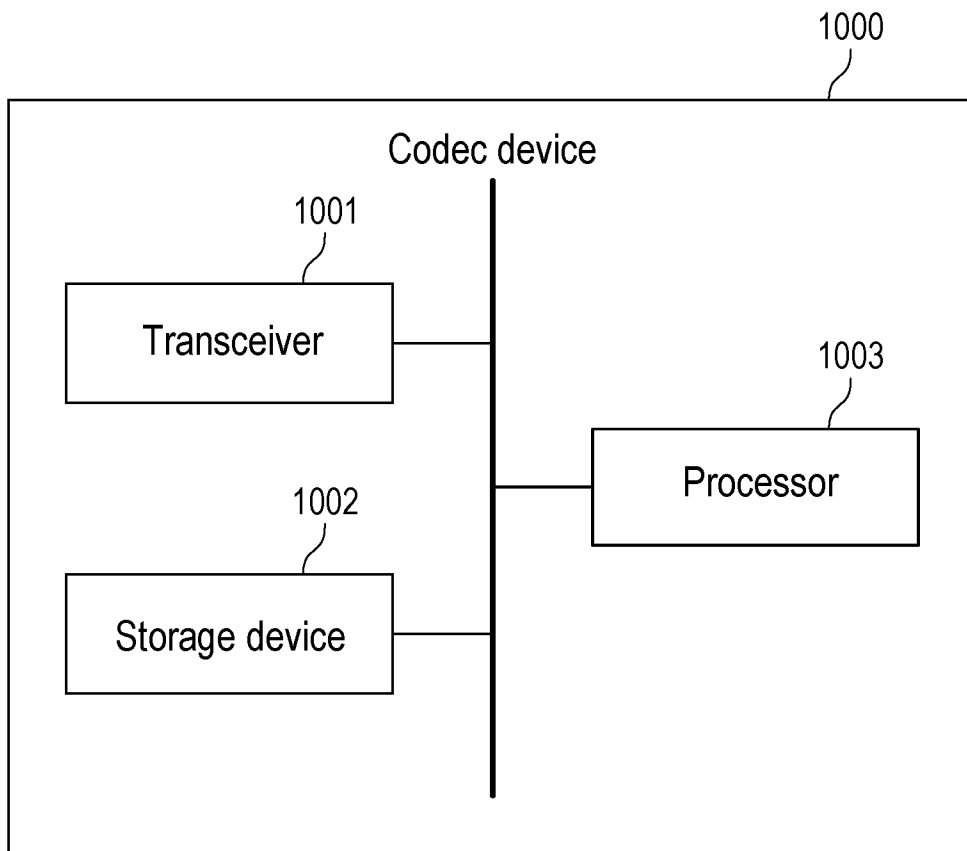


FIG. 24

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 23/32810

A. CLASSIFICATION OF SUBJECT MATTER

IPC - INV. G06T 9/00, H04N 19/136, H04N 19/70 (2023.01)

ADD. H04N 19/10, H04N 19/46 (2023.01)

CPC - INV. G06T 9/00, H04N 19/136, H04N 19/70

ADD. H04N 19/10, H04N 19/46

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)

See Search History document

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

See Search History document

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

See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2021/0297699 A1 (Panasonic Intellectual Property Corporation of America) 23 September 2021 (23.09.2021); entire document, especially, abstract, FIG. 5-7, para [0225], [0227], [0272], [0287], [0418], [0594],	1-33
A	US 2020/0314435 A1 (Apple, Inc.) 01 October 2020 (01.10.2020); entire document, especially, abstract, FIG.2A, para [0112]-[0115], [0121], [0436], [0466]-[0467], [0490], [0497], [0541], [0741]	1-33
A	US 2021/0227232 A1 (LG Electronics, Inc.) 22 July 2021 (22.07.2021); entire document	1-33
A	US 2020/0381022 A1 (Nokia Technologies Oy) 03 December 2020 (03.12.2020); entire document	1-33
P,Y	WO 2023/172703 A1 (Innopeak Technology, Inc.) 14 September 2023 (14.09.2023); entire document, especially, abstract, FIG. 3,8,9, para [0039], [0040], [0051]-[0058]	1-33

☐ 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

"D" document cited by the applicant in the international application

"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

17 November 2023

Date of mailing of the international search report

DEC 12 2023

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents

P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Kari Rodriguez

Telephone No. PCT Helpdesk: 571-272-4300