US 20250259334A1

(19) United States

(12) Patent Application Publication

WANG et al.

(10) Pub. No.: US 2025/0259334 A1

(43) Pub. Date: Aug. 14, 2025

(54) METHOD, APPARATUS, AND MEDIUM FOR POINT CLOUD CODING

(71) Applicants: **Douyin Vision Co., Ltd.**, Beijing (CN); **Bytedance Inc.**, Los Angeles, CA (US)

(72) Inventors: **Wenyi WANG**, Beijing (CN); **Yingzhan XU**, Beijing (CN); **Kai ZHANG**, Los Angeles, CA (US); **Li ZHANG**, Los Angeles, CA (US)

(21) Appl. No.: **19/170,020**

(22) Filed: **Apr. 3, 2025**

Related U.S. Application Data

(63) Continuation of application No. PCT/CN2023/122738, filed on Sep. 28, 2023.

(30) **Foreign Application Priority Data**

Oct. 4, 2022 (WO) PCT/CN2022/123705

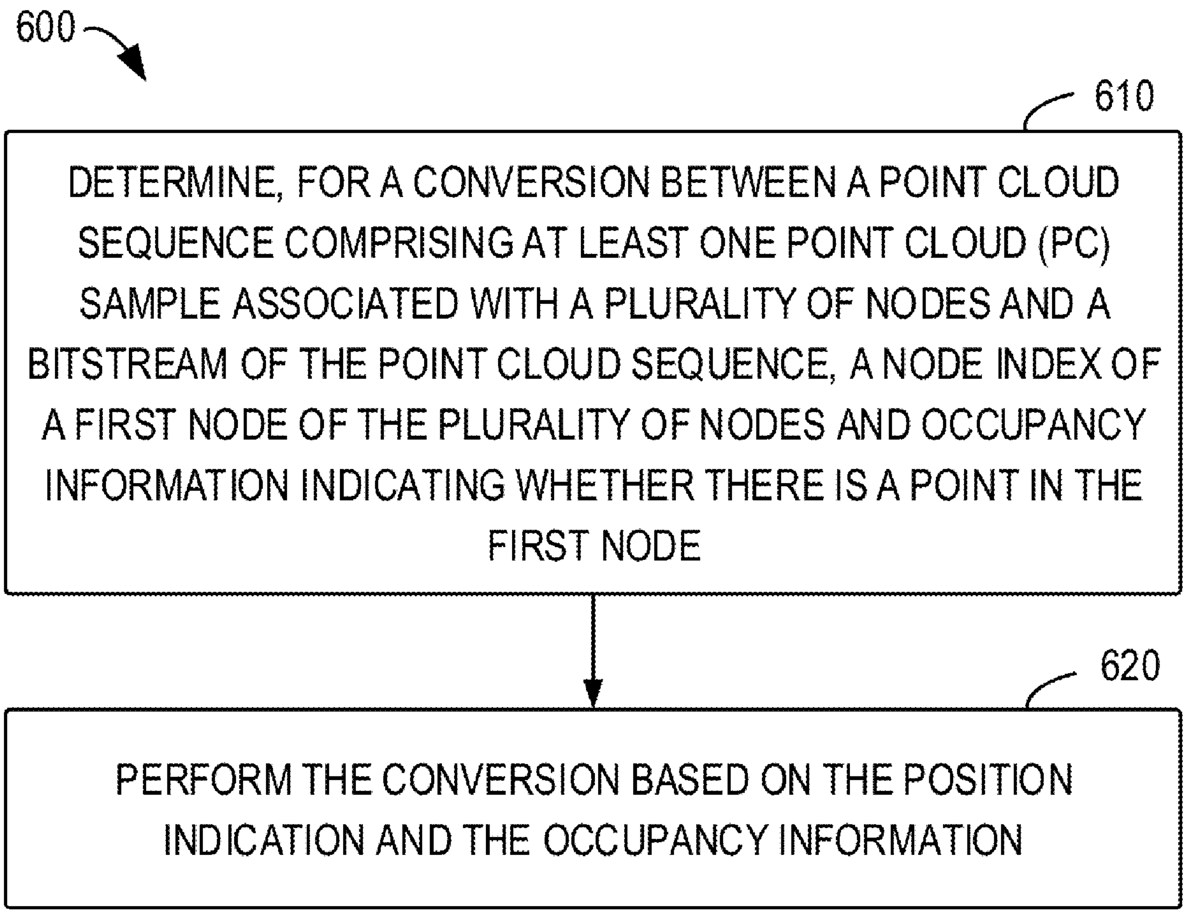
Publication Classification

(51) **Int. Cl.**
G06T 9/00 (2006.01)

(52) **U.S. Cl.**
CPC *G06T 9/00* (2013.01)

(57) **ABSTRACT**

Embodiments of the present disclosure provide a method for point cloud coding. The method comprises: determining, for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with a plurality of nodes and a bitstream of the point cloud sequence, a node index of a first node of the plurality of nodes, wherein the first node is stored in a data structure and is indicated in the data structure by the node index; and performing the conversion based on the position indication and the occupancy information.


600
610
DETERMINE, FOR A CONVERSION BETWEEN A POINT CLOUD SEQUENCE COMPRISING AT LEAST ONE POINT CLOUD (PC) SAMPLE ASSOCIATED WITH A PLURALITY OF NODES AND A BITSTREAM OF THE POINT CLOUD SEQUENCE, A NODE INDEX OF A FIRST NODE OF THE PLURALITY OF NODES AND OCCUPANCY INFORMATION INDICATING WHETHER THERE IS A POINT IN THE FIRST NODE
620
PERFORM THE CONVERSION BASED ON THE POSITION INDICATION AND THE OCCUPANCY INFORMATION

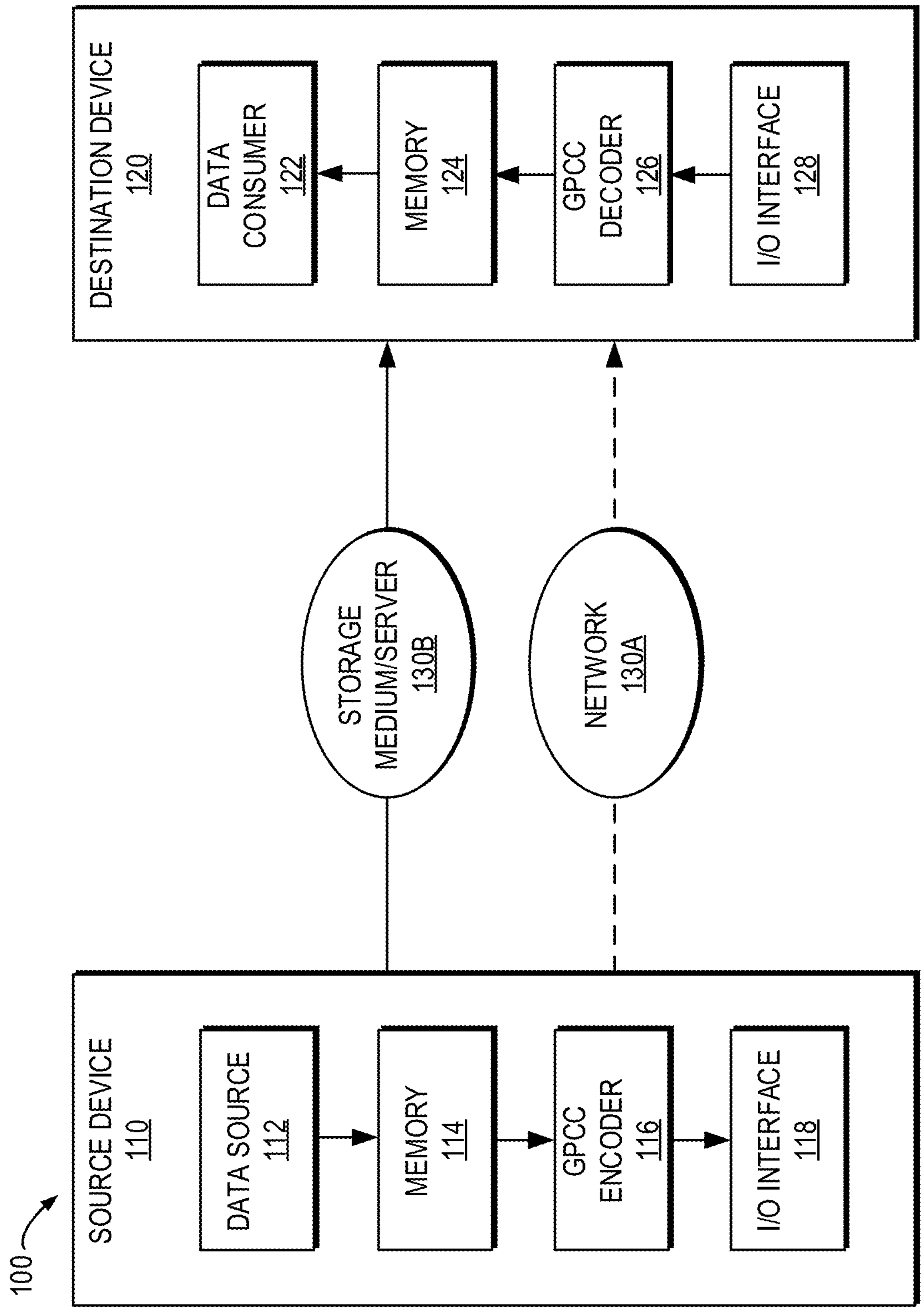

100
SOURCE DEVICE 110
DATA SOURCE 112
MEMORY 114
GPCC ENCODER 116
I/O INTERFACE 118
STORAGE MEDIUM/SERVER 130B
NETWORK 130A
DESTINATION DEVICE 120
DATA CONSUMER 122
MEMORY 124
GPCC DECODER 126
I/O INTERFACE 128

Fig. 1

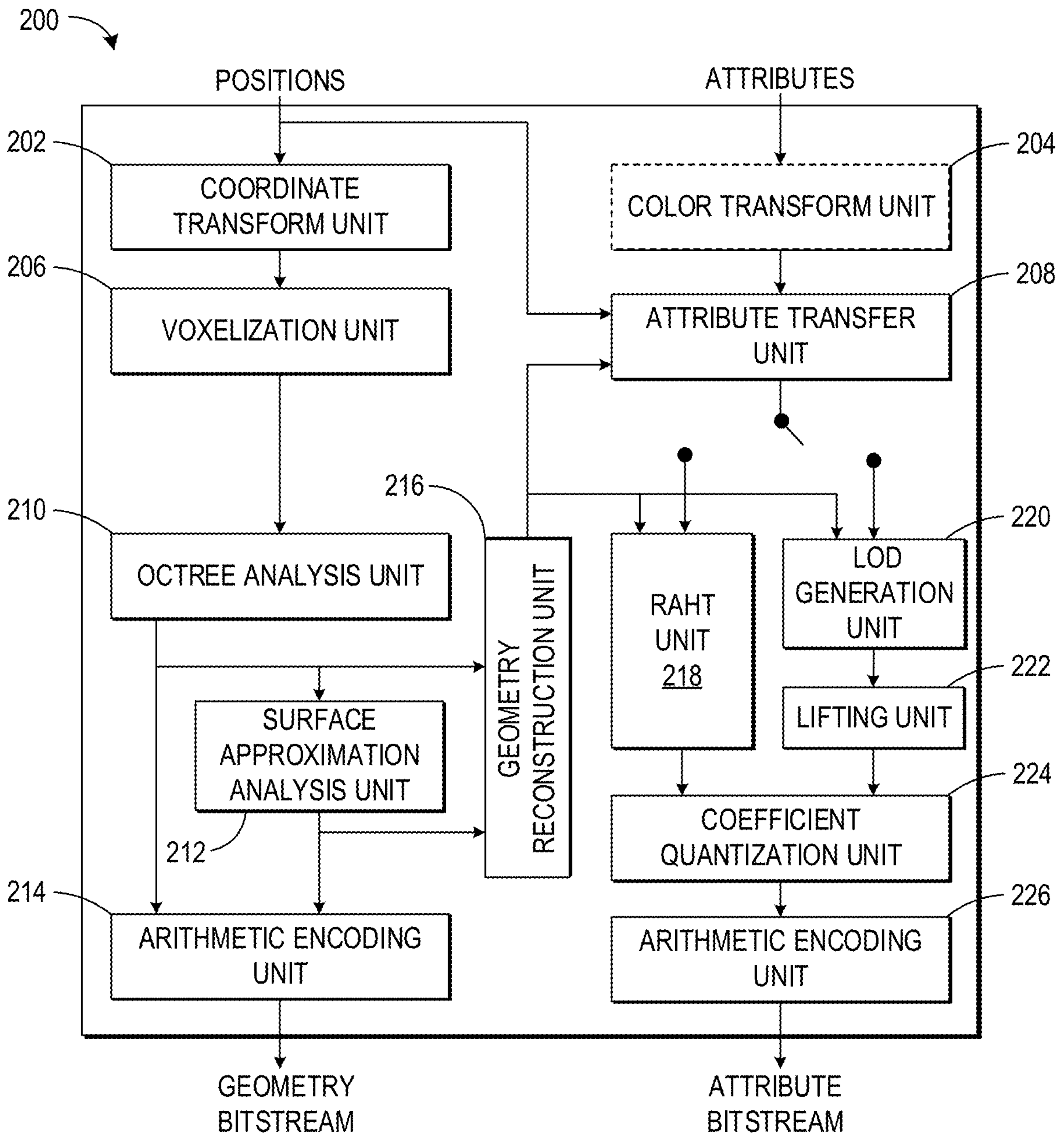

200
POSITIONS
ATTRIBUTES
202
COORDINATE TRANSFORM UNIT
204
COLOR TRANSFORM UNIT
206
VOXELIZATION UNIT
208
ATTRIBUTE TRANSFER UNIT
216
210
OCTREE ANALYSIS UNIT
GEOMETRY RECONSTRUCTION UNIT
RAHT UNIT 218
220
LOD GENERATION UNIT
222
LIFTING UNIT
SURFACE APPROXIMATION ANALYSIS UNIT
212
224
COEFFICIENT QUANTIZATION UNIT
214
ARITHMETIC ENCODING UNIT
226
ARITHMETIC ENCODING UNIT
GEOMETRY BITSTREAM
ATTRIBUTE BITSTREAM

Fig. 2

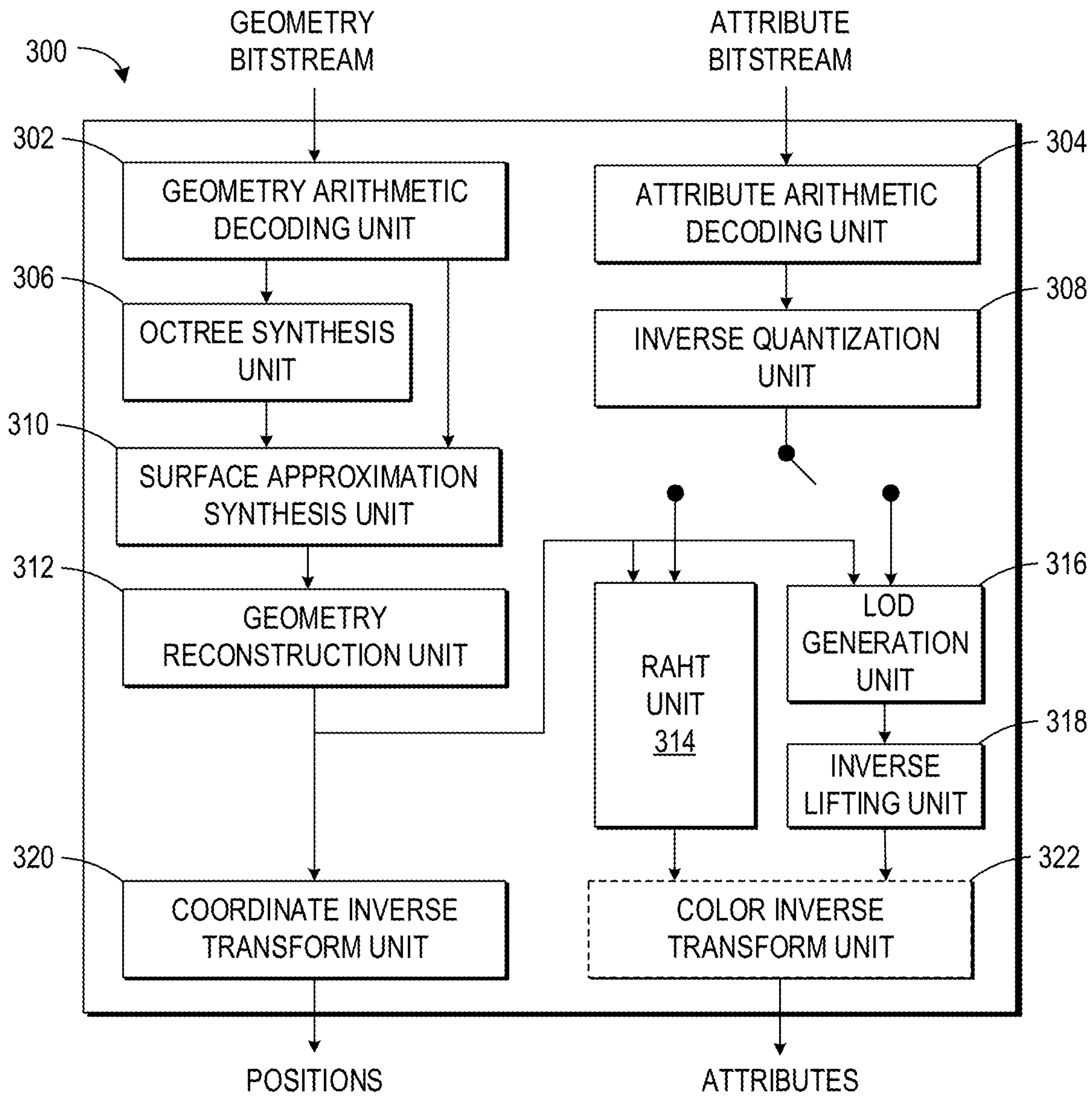

300
GEOMETRY BITSTREAM
ATTRIBUTE BITSTREAM
302
GEOMETRY ARITHMETIC DECODING UNIT
304
ATTRIBUTE ARITHMETIC DECODING UNIT
306
OCTREE SYNTHESIS UNIT
308
INVERSE QUANTIZATION UNIT
310
SURFACE APPROXIMATION SYNTHESIS UNIT
312
GEOMETRY RECONSTRUCTION UNIT
RAHT UNIT 314
316
LOD GENERATION UNIT
318
INVERSE LIFTING UNIT
320
COORDINATE INVERSE TRANSFORM UNIT
322
COLOR INVERSE TRANSFORM UNIT
POSITIONS
ATTRIBUTES

Fig. 3

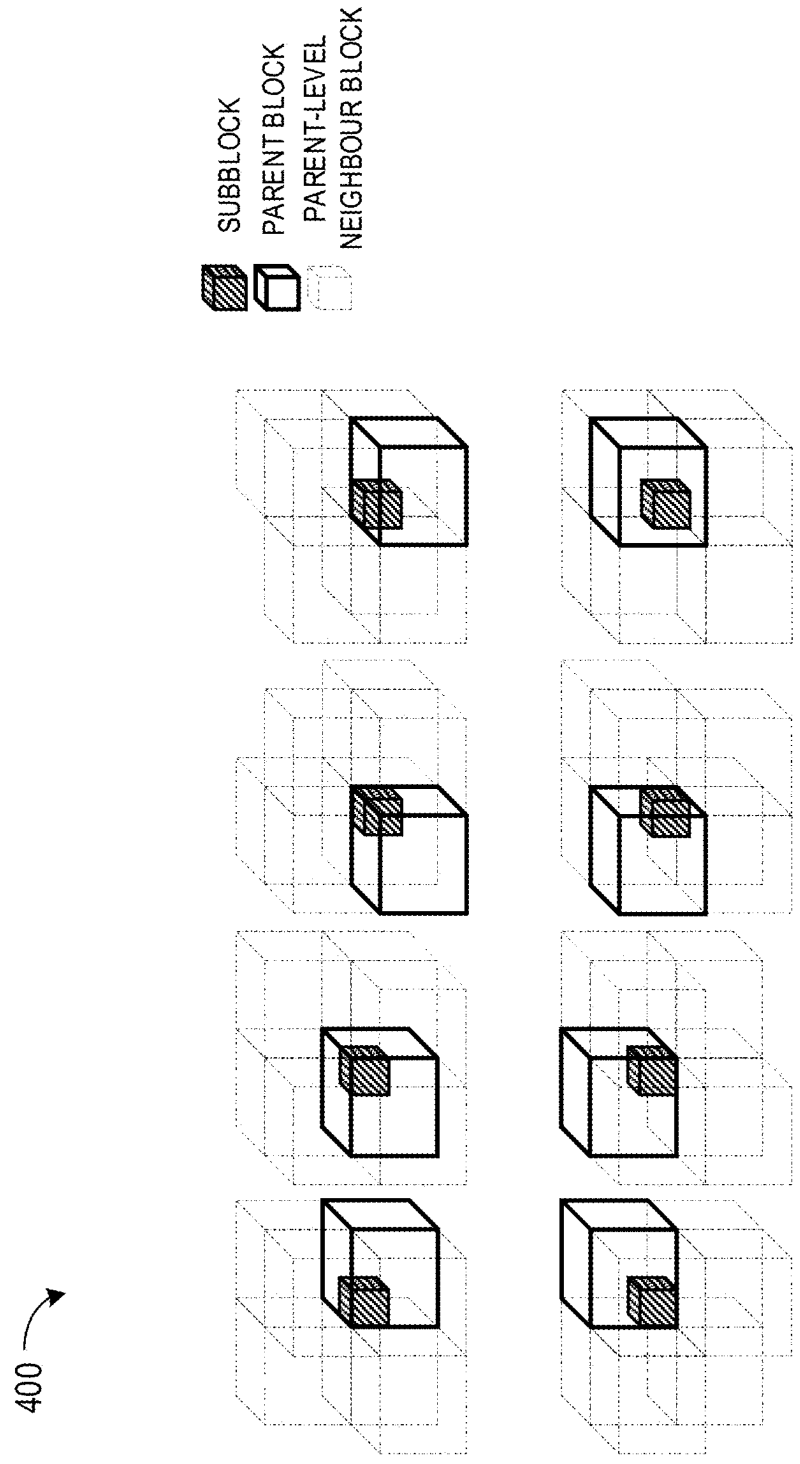


Fig. 4

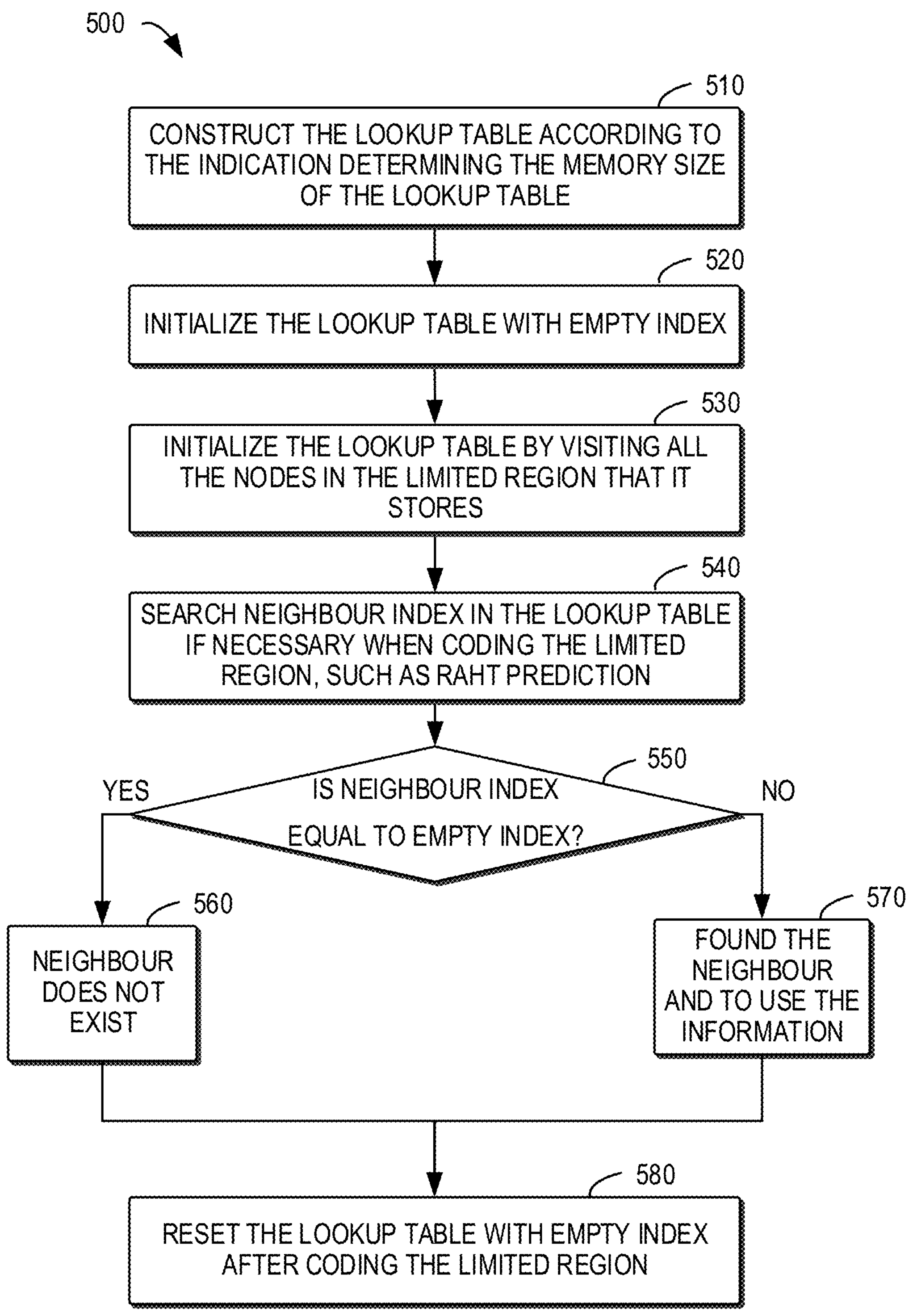

500
510
CONSTRUCT THE LOOKUP TABLE ACCORDING TO THE INDICATION DETERMINING THE MEMORY SIZE OF THE LOOKUP TABLE
520
INITIALIZE THE LOOKUP TABLE WITH EMPTY INDEX
530
INITIALIZE THE LOOKUP TABLE BY VISITING ALL THE NODES IN THE LIMITED REGION THAT IT STORES
540
SEARCH NEIGHBOUR INDEX IN THE LOOKUP TABLE IF NECESSARY WHEN CODING THE LIMITED REGION, SUCH AS RAHT PREDICTION
550
IS NEIGHBOUR INDEX EQUAL TO EMPTY INDEX?
YES
NO
560
NEIGHBOUR DOES NOT EXIST
570
FOUND THE NEIGHBOUR AND TO USE THE INFORMATION
580
RESET THE LOOKUP TABLE WITH EMPTY INDEX AFTER CODING THE LIMITED REGION

Fig. 5

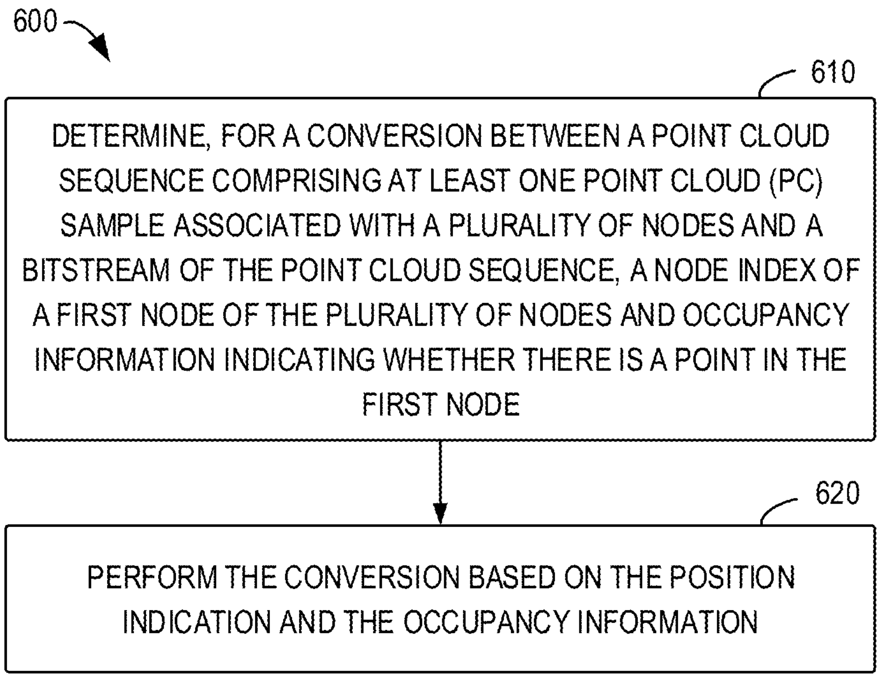

600
610
DETERMINE, FOR A CONVERSION BETWEEN A POINT CLOUD SEQUENCE COMPRISING AT LEAST ONE POINT CLOUD (PC) SAMPLE ASSOCIATED WITH A PLURALITY OF NODES AND A BITSTREAM OF THE POINT CLOUD SEQUENCE, A NODE INDEX OF A FIRST NODE OF THE PLURALITY OF NODES AND OCCUPANCY INFORMATION INDICATING WHETHER THERE IS A POINT IN THE FIRST NODE
620
PERFORM THE CONVERSION BASED ON THE POSITION INDICATION AND THE OCCUPANCY INFORMATION

Fig. 6

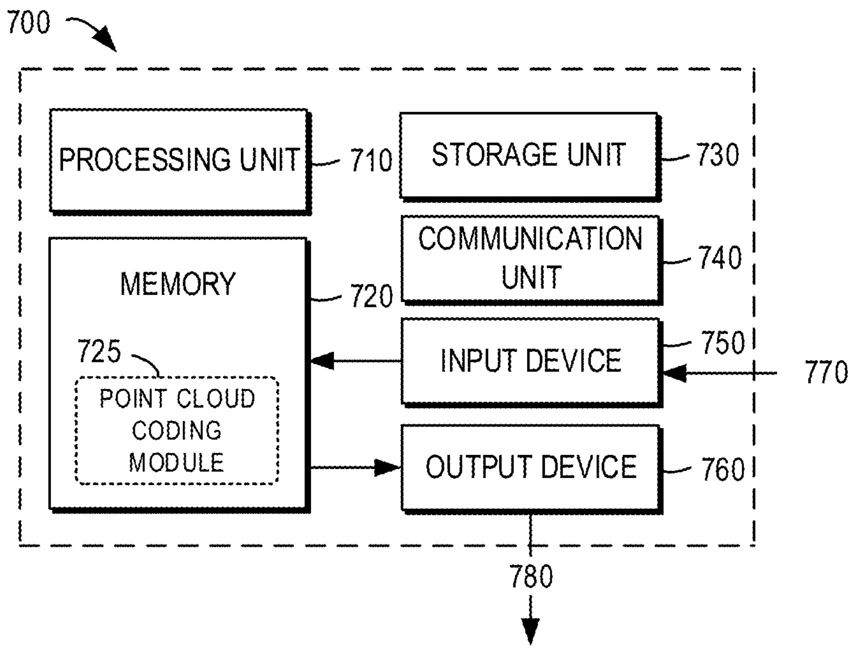

700
PROCESSING UNIT
710
STORAGE UNIT
730
COMMUNICATION UNIT
740
MEMORY
720
725
POINT CLOUD CODING MODULE
INPUT DEVICE
750
770
OUTPUT DEVICE
760
780

Fig. 7

METHOD, APPARATUS, AND MEDIUM FOR POINT CLOUD CODING

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation of International Application No. PCT/CN2023/122738, filed on Sep. 28, 2023, which claims the benefit of International Application No. PCT/CN2022/123705 filed on Oct. 4, 2022. The entire contents of these applications are hereby incorporated by reference in their entireties.

FIELDS

[0002] Embodiments of the present disclosure relates generally to point cloud coding techniques, and more particularly, to a lookup table for neighbour search in region-adaptive hierarchical transform (RAHT).

BACKGROUND

[0003] A point cloud is a collection of individual data points in a three-dimensional (3D) plane with each point having a set coordinate on the X, Y, and Z axes. Thus, a point cloud may be used to represent the physical content of the three-dimensional space. Point clouds have shown to be a promising way to represent 3D visual data for a wide range of immersive applications, from augmented reality to autonomous cars.

[0004] Point cloud coding standards have evolved primarily through the development of the well-known MPEG organization. MPEG, short for Moving Picture Experts Group, is one of the main standardization groups dealing with multimedia. In 2017, the MPEG 3D Graphics Coding group (3DG) published a call for proposals (CFP) document to start to develop point cloud coding standard. The final standard will consist in two classes of solutions. Video-based Point Cloud Compression (V-PCC or VPCC) is appropriate for point sets with a relatively uniform distribution of points. Geometry-based Point Cloud Compression (G-PCC or GPCC) is appropriate for more sparse distributions. However, coding efficiency of conventional point cloud coding techniques is generally expected to be further improved.

SUMMARY

[0005] Embodiments of the present disclosure provide a solution for point cloud coding.

[0006] In a first aspect, a method for point cloud coding is proposed. The method comprises: determining, for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with a plurality of nodes and a bitstream of the point cloud sequence, a node index of a first node of the plurality of nodes, wherein the first node is stored in a data structure and is indicated in the data structure by the node index; and performing the conversion based on the position indication and the occupancy information.

[0007] In a second aspect, an apparatus for point cloud coding is proposed. The apparatus comprises a processor and a non-transitory memory with instructions thereon. The instructions upon execution by the processor, cause the processor to perform a method in accordance with the first aspect of the present disclosure.

[0008] In a third aspect, a non-transitory computer-readable storage medium is proposed. The non-transitory computer-readable storage medium stores instructions that cause a processor to perform a method in accordance with the first aspect of the present disclosure.

[0009] In a fourth aspect, another non-transitory computer-readable recording medium is proposed. The non-transitory computer-readable recording medium stores a bitstream of a point cloud sequence which is generated by a method performed by an apparatus for point cloud coding. The method comprises: determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; and generating the bitstream based on the position indication and the occupancy information.

[0010] In a fifth aspect, a method for storing a bitstream of a point cloud sequence is proposed. The method comprises: determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; generating the bitstream based on the position indication and the occupancy information; and storing the bitstream in a non-transitory computer-readable recording medium.

[0011] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Through the following detailed description with reference to the accompanying drawings, the above and other objectives, features, and advantages of example embodiments of the present disclosure will become more apparent. In the example embodiments of the present disclosure, the same reference numerals usually refer to the same components.

[0013] FIG. **1** is a block diagram that illustrates an example point cloud coding system that may utilize the techniques of the present disclosure;

[0014] FIG. **2** illustrates a block diagram that illustrates an example point cloud encoder in accordance with some embodiments of the present disclosure;

[0015] FIG. **3** illustrates a block diagram that illustrates an example point cloud decoder in accordance with some embodiments of the present disclosure;

[0016] FIG. **4** illustrates an example diagram of parent-level nodes for each sub-node of transform unit node in accordance with some embodiments of the present disclosure;

[0017] FIG. **5** illustrates an example diagram of construction of a lookup table in accordance with some embodiments of the present disclosure;

[0018] FIG. **6** illustrates a flowchart of a method for point cloud coding in accordance with embodiments of the present disclosure; and

[0019] FIG. **7** illustrates a block diagram of a computing device in which various embodiments of the present disclosure can be implemented.

[0020] Throughout the drawings, the same or similar reference numerals usually refer to the same or similar elements.

DETAILED DESCRIPTION

[0021] Principle of the present disclosure will now be described with reference to some embodiments. It is to be understood that these embodiments are described only for the purpose of illustration and help those skilled in the art to understand and implement the present disclosure, without suggesting any limitation as to the scope of the disclosure. The disclosure described herein can be implemented in various manners other than the ones described below.

[0022] In the following description and claims, unless defined otherwise, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skills in the art to which this disclosure belongs.

[0023] References in the present disclosure to "one embodiment," "an embodiment," "an example embodiment," and the like indicate that the embodiment described may include a particular feature, structure, or characteristic, but it is not necessary that every embodiment includes the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an example embodiment, it is submitted that it is within the knowledge of one skilled in the art to affect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0024] It shall be understood that although the terms "first" and "second" etc. may be used herein to describe various elements, these elements should not be limited by these terms. These terms are only used to distinguish one element from another. For example, a first element could be termed a second element, and similarly, a second element could be termed a first element, without departing from the scope of example embodiments. As used herein, the term "and/or" includes any and all combinations of one or more of the listed terms.

[0025] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of example embodiments. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises", "comprising", "has", "having", "includes" and/or "including", when used herein, specify the presence of stated features, elements, and/or components etc., but do not preclude the presence or addition of one or more other features, elements, components and/or combinations thereof.

Example Environment

[0026] FIG. **1** is a block diagram that illustrates an example point cloud coding system **100** that may utilize the techniques of the present disclosure. As shown, the point cloud coding system **100** may include a source device **110** and a destination device **120**. The source device **110** can be also referred to as a point cloud encoding device, and the destination device **120** can be also referred to as a point cloud decoding device. In operation, the source device **110** can be configured to generate encoded point cloud data and the destination device **120** can be configured to decode the encoded point cloud data generated by the source device **110**. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) point cloud data, i.e., to support point cloud compression. The coding may be effective in compressing and/or decompressing point cloud data.

[0027] Source device **100** and destination device **120** may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as smartphones and mobile phones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming devices, vehicles (e.g., terrestrial or marine vehicles, spacecraft, aircraft, etc.), robots, LIDAR devices, satellites, extended reality devices, or the like. In some cases, source device **100** and destination device **120** may be equipped for wireless communication.

[0028] The source device **100** may include a data source **112**, a memory **114**, a GPCC encoder **116**, and an input/output (I/O) interface **118**. The destination device **120** may include an input/output (I/O) interface **128**, a GPCC decoder **126**, a memory **124**, and a data consumer **122**. In accordance with this disclosure, GPCC encoder **116** of source device **100** and GPCC decoder **126** of destination device **120** may be configured to apply the techniques of this disclosure related to point cloud coding. Thus, source device **100** represents an example of an encoding device, while destination device **120** represents an example of a decoding device. In other examples, source device **100** and destination device **120** may include other components or arrangements. For example, source device **100** may receive data (e.g., point cloud data) from an internal or external source. Likewise, destination device **120** may interface with an external data consumer, rather than include a data consumer in the same device.

[0029] In general, data source **112** represents a source of point cloud data (i.e., raw, unencoded point cloud data) and may provide a sequential series of "frames" of the point cloud data to GPCC encoder **116**, which encodes point cloud data for the frames. In some examples, data source **112** generates the point cloud data. Data source **112** of source device **100** may include a point cloud capture device, such as any of a variety of cameras or sensors, e.g., one or more video cameras, an archive containing previously captured point cloud data, a 3D scanner or a light detection and ranging (LIDAR) device, and/or a data feed interface to receive point cloud data from a data content provider. Thus, in some examples, data source **112** may generate the point cloud data based on signals from a LIDAR apparatus. Alternatively or additionally, point cloud data may be computer-generated from scanner, camera, sensor or other data. For example, data source **112** may generate the point cloud data, or produce a combination of live point cloud data, archived point cloud data, and computer-generated point cloud data. In each case, GPCC encoder **116** encodes the captured, pre-captured, or computer-generated point cloud data. GPCC encoder **116** may rearrange frames of the point cloud data from the received order (sometimes referred to as "display order") into a coding order for coding. GPCC encoder **116** may generate one or more bitstreams including encoded point cloud data. Source device **100** may then output the encoded point cloud data via I/O interface **118** for reception and/or retrieval by, e.g., I/O interface **128** of destination device **120**. The encoded point cloud data may be transmitted directly to destination device **120** via the I/O interface **118** through the network **130**A. The encoded point cloud data may also be stored onto a storage medium/server **130**B for access by destination device **120**.

[0030] Memory **114** of source device **100** and memory **124** of destination device **120** may represent general purpose memories. In some examples, memory **114** and memory **124** may store raw point cloud data, e.g., raw point cloud data from data source **112** and raw, decoded point cloud data from GPCC decoder **126**. Additionally or alternatively, memory **114** and memory **124** may store software instructions executable by, e.g., GPCC encoder **116** and GPCC decoder **126**, respectively. Although memory **114** and memory **124** are shown separately from GPCC encoder **116** and GPCC decoder **126** in this example, it should be understood that GPCC encoder **116** and GPCC decoder **126** may also include internal memories for functionally similar or equivalent purposes. Furthermore, memory **114** and memory **124** may store encoded point cloud data, e.g., output from GPCC encoder **116** and input to GPCC decoder **126**. In some examples, portions of memory **114** and memory **124** may be allocated as one or more buffers, e.g., to store raw, decoded, and/or encoded point cloud data. For instance, memory **114** and memory **124** may store point cloud data.

[0031] I/O interface **118** and I/O interface **128** may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where I/O interface **118** and I/O interface **128** comprise wireless components, I/O interface **118** and I/O interface **128** may be configured to transfer data, such as encoded point cloud data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where I/O interface **118** comprises a wireless transmitter, I/O interface **118** and I/O interface **128** may be configured to transfer data, such as encoded point cloud data, according to other wireless standards, such as an IEEE 802.11 specification. In some examples, source device **100** and/or destination device **120** may include respective system-on-a-chip (SoC) devices. For example, source device **100** may include an SoC device to perform the functionality attributed to GPCC encoder **116** and/or I/O interface **118**, and destination device **120** may include an SoC device to perform the functionality attributed to GPCC decoder **126** and/or I/O interface **128**.

[0032] The techniques of this disclosure may be applied to encoding and decoding in support of any of a variety of applications, such as communication between autonomous vehicles, communication between scanners, cameras, sensors and processing devices such as local or remote servers, geographic mapping, or other applications.

[0033] I/O interface **128** of destination device **120** receives an encoded bitstream from source device **110**. The encoded bitstream may include signaling information defined by GPCC encoder **116**, which is also used by GPCC decoder **126**, such as syntax elements having values that represent a point cloud. Data consumer **122** uses the decoded data. For example, data consumer **122** may use the decoded point cloud data to determine the locations of physical objects. In some examples, data consumer **122** may comprise a display to present imagery based on the point cloud data.

[0034] GPCC encoder **116** and GPCC decoder **126** each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of GPCC encoder **116** and GPCC decoder **126** may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including GPCC encoder **116** and/or GPCC decoder **126** may comprise one or more integrated circuits, microprocessors, and/or other types of devices.

[0035] GPCC encoder **116** and GPCC decoder **126** may operate according to a coding standard, such as video point cloud compression (VPCC) standard or a geometry point cloud compression (GPCC) standard. This disclosure may generally refer to coding (e.g., encoding and decoding) of frames to include the process of encoding or decoding data. An encoded bitstream generally includes a series of values for syntax elements representative of coding decisions (e.g., coding modes).

[0036] A point cloud may contain a set of points in a 3D space, and may have attributes associated with the point. The attributes may be color information such as R, G, B or Y, Cb, Cr, or reflectance information, or other attributes. Point clouds may be captured by a variety of cameras or sensors such as LIDAR sensors and 3D scanners and may also be computer-generated. Point cloud data are used in a variety of applications including, but not limited to, construction (modeling), graphics (3D models for visualizing and animation), and the automotive industry (LIDAR sensors used to help in navigation).

[0037] FIG. **2** is a block diagram illustrating an example of a GPCC encoder **200**, which may be an example of the GPCC encoder **116** in the system **100** illustrated in FIG. **1**, in accordance with some embodiments of the present disclosure. FIG. **3** is a block diagram illustrating an example of a GPCC decoder **300**, which may be an example of the GPCC decoder **126** in the system **100** illustrated in FIG. **1**, in accordance with some embodiments of the present disclosure.

[0038] In both GPCC encoder **200** and GPCC decoder **300**, point cloud positions are coded first. Attribute coding depends on the decoded geometry. In FIG. **2** and FIG. **3**, the region adaptive hierarchical transform (RAHT) unit **218**, surface approximation analysis unit **212**, RAHT unit **314** and surface approximation synthesis unit **310** are options typically used for Category 1 data. The level-of-detail (LOD) generation unit **220**, lifting unit **222**, LOD generation unit **316** and inverse lifting unit **318** are options typically used for Category 3 data. All the other units are common between Categories 1 and 3.

[0039] For Category 3 data, the compressed geometry is typically represented as an octree from the root all the way down to a leaf level of individual voxels. For Category 1 data, the compressed geometry is typically represented by a pruned octree (i.e., an octree from the root down to a leaf level of blocks larger than voxels) plus a model that approximates the surface within each leaf of the pruned octree. In this way, both Category 1 and 3 data share the octree coding mechanism, while Category 1 data may in addition approximate the voxels within each leaf with a surface model. The surface model used is a triangulation comprising 1-10 triangles per block, resulting in a triangle soup. The Category 1 geometry codec is therefore known as the Trisoup geometry codec, while the Category 3 geometry codec is known as the Octree geometry codec.

[0040] In the example of FIG. **2**, GPCC encoder **200** may include a coordinate transform unit **202**, a color transform unit **204**, a voxelization unit **206**, an attribute transfer unit **208**, an octree analysis unit **210**, a surface approximation analysis unit **212**, an arithmetic encoding unit **214**, a geometry reconstruction unit **216**, an RAHT unit **218**, a LOD generation unit **220**, a lifting unit **222**, a coefficient quantization unit **224**, and an arithmetic encoding unit **226**.

[0041] As shown in the example of FIG. **2**, GPCC encoder **200** may receive a set of positions and a set of attributes. The positions may include coordinates of points in a point cloud. The attributes may include information about points in the point cloud, such as colors associated with points in the point cloud.

[0042] Coordinate transform unit **202** may apply a transform to the coordinates of the points to transform the coordinates from an initial domain to a transform domain. This disclosure may refer to the transformed coordinates as transform coordinates. Color transform unit **204** may apply a transform to convert color information of the attributes to a different domain. For example, color transform unit **204** may convert color information from an RGB color space to a YCbCr color space.

[0043] Furthermore, in the example of FIG. **2**, voxelization unit **206** may voxelize the transform coordinates. Voxelization of the transform coordinates may include quantizing and removing some points of the point cloud. In other words, multiple points of the point cloud may be subsumed within a single "voxel," which may thereafter be treated in some respects as one point. Furthermore, octree analysis unit **210** may generate an octree based on the voxelized transform coordinates. Additionally, in the example of FIG. **2**, surface approximation analysis unit **212** may analyze the points to potentially determine a surface representation of sets of the points. Arithmetic encoding unit **214** may perform arithmetic encoding on syntax elements representing the information of the octree and/or surfaces determined by surface approximation analysis unit **212**. GPCC encoder **200** may output these syntax elements in a geometry bitstream.

[0044] Geometry reconstruction unit **216** may reconstruct transform coordinates of points in the point cloud based on the octree, data indicating the surfaces determined by surface approximation analysis unit **212**, and/or other information. The number of transform coordinates reconstructed by geometry reconstruction unit **216** may be different from the original number of points of the point cloud because of voxelization and surface approximation. This disclosure may refer to the resulting points as reconstructed points. Attribute transfer unit **208** may transfer attributes of the original points of the point cloud to reconstructed points of the point cloud data.

[0045] Furthermore, RAHT unit **218** may apply RAHT coding to the attributes of the reconstructed points. Alternatively or additionally, LOD generation unit **220** and lifting unit **222** may apply LOD processing and lifting, respectively, to the attributes of the reconstructed points. RAHT unit **218** and lifting unit **222** may generate coefficients based on the attributes. Coefficient quantization unit **224** may quantize the coefficients generated by RAHT unit **218** or lifting unit **222**. Arithmetic encoding unit **226** may apply arithmetic coding to syntax elements representing the quantized coefficients. GPCC encoder **200** may output these syntax elements in an attribute bitstream.

[0046] In the example of FIG. **3**, GPCC decoder **300** may include a geometry arithmetic decoding unit **302**, an attribute arithmetic decoding unit **304**, an octree synthesis unit **306**, an inverse quantization unit **308**, a surface approximation synthesis unit **310**, a geometry reconstruction unit **312**, a RAHT unit **314**, a LOD generation unit **316**, an inverse lifting unit **318**, a coordinate inverse transform unit **320**, and a color inverse transform unit **322**.

[0047] GPCC decoder **300** may obtain a geometry bitstream and an attribute bitstream. Geometry arithmetic decoding unit **302** of decoder **300** may apply arithmetic decoding (e.g., CABAC or other type of arithmetic decoding) to syntax elements in the geometry bitstream. Similarly, attribute arithmetic decoding unit **304** may apply arithmetic decoding to syntax elements in attribute bitstream.

[0048] Octree synthesis unit **306** may synthesize an octree based on syntax elements parsed from geometry bitstream. In instances where surface approximation is used in geometry bitstream, surface approximation synthesis unit **310** may determine a surface model based on syntax elements parsed from geometry bitstream and based on the octree.

[0049] Furthermore, geometry reconstruction unit **312** may perform a reconstruction to determine coordinates of points in a point cloud. Coordinate inverse transform unit **320** may apply an inverse transform to the reconstructed coordinates to convert the reconstructed coordinates (positions) of the points in the point cloud from a transform domain back into an initial domain.

[0050] Additionally, in the example of FIG. **3**, inverse quantization unit **308** may inverse quantize attribute values. The attribute values may be based on syntax elements obtained from attribute bitstream (e.g., including syntax elements decoded by attribute arithmetic decoding unit **304**).

[0051] Depending on how the attribute values are encoded, RAHT unit **314** may perform RAHT coding to determine, based on the inverse quantized attribute values, color values for points of the point cloud. Alternatively, LOD generation unit **316** and inverse lifting unit **318** may determine color values for points of the point cloud using a level of detail-based technique.

[0052] Furthermore, in the example of FIG. **3**, color inverse transform unit **322** may apply an inverse color transform to the color values. The inverse color transform may be an inverse of a color transform applied by color transform unit **204** of encoder **200**. For example, color transform unit **204** may transform color information from an RGB color space to a YCbCr color space. Accordingly, color inverse transform unit **322** may transform color information from the YCbCr color space to the RGB color space.

[0053] The various units of FIG. **2** and FIG. **3** are illustrated to assist with understanding the operations performed by encoder **200** and decoder **300**. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0054] Some exemplary embodiments of the present disclosure will be described in detailed hereinafter. It should be understood that section headings are used in the present document to facilitate ease of understanding and do not limit the embodiments disclosed in a section to only that section. Furthermore, while certain embodiments are described with reference to GPCC or other specific point cloud codecs, the disclosed techniques are applicable to other point cloud coding technologies also. Furthermore, while some embodiments describe point cloud coding steps in detail, it will be understood that corresponding steps decoding that undo the coding will be implemented by a decoder.

1. Brief Summary

[0055] This disclosure is related to point cloud coding technologies. Specifically, it is related to lookup table for neighbour search in region-adaptive hierarchical transform (RAHT). The ideas may be applied individually or in various combination, to any point cloud coding standard or non-standard point cloud codec, e.g., the being-developed Geometry based Point Cloud Compression (G-PCC) and Low Latency Low Complexity Codec (L3C2).

2. Abbreviations

[0056] G-PCC Geometry based Point Cloud Compression

[0057] L3C2 Low Latency Low Complexity Codec

[0058] MPEG Moving Picture Experts Group

[0059] 3DG 3D Graphics Coding Group

[0060] CFP Call For Proposal

[0061] V-PCC Video-based Point Cloud Compression

[0062] RAHT Region-Adaptive Hierarchical Transform

[0063] DC Direct Current

[0064] AC Alternating Current

3. Introduction

[0065] MPEG, short for Moving Picture Experts Group, is one of the main standardization groups dealing with multimedia. In 2017, the MPEG 3D Graphics Coding group (3DG) published a call for proposals (CFP) document to start to develop point cloud coding standard. The final standard will consist in two classes of solutions. Video-based Point Cloud Compression (V-PCC) is appropriate for point sets with a relatively uniform distribution of points. Geometry-based Point Cloud Compression (G-PCC) is appropriate for more sparse distributions. Both V-PCC and G-PCC support the coding and decoding for single point cloud and point cloud sequence.

[0066] In one point cloud, there may be geometry information and attribute information. Geometry information is used to describe the geometry locations of the data points. Attribute information is used to record some details of the data points, such as textures, normal vectors, reflections and so on.

3.1 Octree Geometry Compression

[0067] Point cloud codec can process the various information in different ways. Usually there are many optional tools in the codec to support the coding and decoding of geometry information and attribute information respectively. Among geometry coding tools in G-PCC, octree geometry compression has an important influence for point cloud geometry coding performance.

[0068] In G-PCC, one of important point cloud geometry coding tools is octree geometry compression, which leverages point cloud geometry spatial correlation. If geometry coding tools is enable, a cubical axis-aligned bounding box, associated with octree root node, will be determined according to point cloud geometry information. Then the bounding box will be subdivided into 8 sub-cubes, which are associated with 8 sub-nodes of root node (a cube is equivalent to node hereafter). An 8-bit code is then generated by specific order to indicate whether the 8 sub-nodes contain points separately, where one bit is associated with one sub-node. The bit associated with one sub-node is named occupancy bit and the 8-bit code generated is named occupancy code. The generated occupancy code will be signaled according to the occupancy information of neighbor node. Then only the nodes which contain points will be subdivided into 8 sub-nodes furtherly. The process will performed recursively until the node size is 1. So, the point cloud geometry information is converted into occupancy code sequences.

[0069] In decoder side, occupancy code sequences will be decoded and the point cloud geometry information can be reconstructed according to the occupancy code sequences.

[0070] A breadth-first scanning order will be used for the octree. In one level of the octree, the octree node will be scanned in a Morton order. If the coordinate of one node is represented by N bits, the coordinate (X, Y, Z) of the node can be represented as follows.

$$X = (x_{N-1}x_{N-2} \ \ldots \ x_1x_0)$$

$$Y = (y_{N-1}y_{N-2} \ \ldots \ y_1y_0)$$

$$Z = (z_{N-1}z_{N-2} \ \ldots \ z_1z_0)$$

[0071] Its Morton code can be represented as follows.

$$M = (x_{N-1}y_{N-1}z_{N-1}x_{N-2}y_{N-2}z_{N-2} \ \ldots \ x_1y_1z_1x_0y_0z_0)$$

[0072] The Morton order is the order from small to large or from large to small according to Morton code.

3.2 Region-Adaptive Hierarchical Transform

[0073] In G-PCC, one of important point cloud attribute coding tools is RAHT. It is a transform that uses the attributes associated with a node in a lower level of the octree to predict the attributes of the nodes in the next level.

It assumes that the positions of the points are given at both the encoder and decoder. RAHT follows the octree scan backwards, from leaf nodes to root node, at each step recombining nodes into larger ones until reaching the root node. At each level of octree, the nodes are processed in the Morton order. At each decomposition, instead of grouping eight nodes at a time, RAHT does it in three steps along each dimension, (e.g., along z, then y then x). If there are L levels in octree, RAHT takes 3L levels to traverse the tree backwards.

[0074] Let the nodes at level l be $g_{l,x,y,z}$, for x, y, z integers. $g_{l,x,y,z}$ was obtained by grouping $g_{l+1,2x,y,z}$ and $g_{l+1,2x+1,y,z}$, where the grouping along the first dimension was an example. RAHT only process occupied nodes. If one of the nodes in the pair is unoccupied, the other one is promoted to the next level, unprocessed, i.e., $g_{l-1,x,y,z}=g_{l,2x,y,z}$ if the latter is the occupied node of the pair. The grouping process is repeated until getting to the root. Note that the grouping process generates nodes at lower levels that are the result of grouping different numbers of voxels along the way. The number of nodes grouped to generate node $g_{l,x,y,z}$ is the weight $\omega_{l,x,y,z}$ of that node. At every grouping of two nodes, say $g_{l,2x,y,z}$ and $g_{l,2x+1,y,z}$, with their respective weights, $\omega_{l,2x,y,z}$ and $\omega_{l,2x+l,y,z}$, RAHT apply the following transform:

$$\begin{bmatrix} g_{l-1,x,y,z} \\ h_{l-1,x,y,z} \end{bmatrix} = T_{\omega_1\omega_2}\begin{bmatrix} g_{l,2x,y,z} \\ h_{l,2x+1,y,z} \end{bmatrix},$$

where $w_i - w_{l,2x,y,z}$ and $w_2 - w_{l,2x+,y,z}$ and

$$T_{\omega_1\omega_2} = \frac{1}{\sqrt{\omega_1+\omega_2}}\begin{bmatrix} \sqrt{\omega_1} & \sqrt{\omega_2} \\ -\sqrt{\omega_2} & \sqrt{\omega_1} \end{bmatrix}.$$

[0075] Note that the transform matrix changes at all times, adapting to the weights, i.e., adapting to the number of leaf nodes that each $g_{l,x,y,z}$ actually represents. The quantities $g_{l,x,y,z}$ are used to group and compose further nodes at a lower level. $h_{l,x,y,z}$ are the actual high-pass coefficients generated by the transform to be encoded and transmitted. Furthermore, weights accumulate for the level above. In the above example,

$$\omega_{l-1,2,y,z} = \omega_{l,2x,y,z} + \omega_{l,2x+1,y,z}.$$

[0076] In the last stage, the tree root, the remaining two voxels $g_{1,0,0,0}$ and $g_{1,1,0,0}$ are transformed into the final two coefficients as:

$$\begin{bmatrix} g_{DC} \\ h_{0,0,0,0} \end{bmatrix} = T_{\omega_{1,0,0,0}\omega_{1,1,0,0}}\begin{bmatrix} g_{1,0,0,0} \\ g_{1,1,0,0} \end{bmatrix},$$

where $g_{DC}=g_{0,0,0,0}$.

3.2.1 Upsampled Transform Domain Prediction

[0077] The transform domain prediction is introduced to improve coding efficiency on RAHT. It is formed of two parts. Firstly, the RAHT tree traversal is changed to be descent based from the previous ascent approach, i.e., a tree of attribute and weight sums is constructed and then RAHT is performed from the root level of the tree to the leaves level for both the encoder and the decoder. In each level, the node is visited in Morton order. The transform is performed in node that has 2×2×2 sub-nodes which is in the next level. The node in which transform is performed may be called as transform node.

[0078] Secondly, for each sub-node of transform node, a corresponding prediction attribute is produced by upsampling the attribute of previous transform level. Actually, only sub-node that contains at last one point will produce a corresponding prediction attribute. The transform node that contains prediction attributes is transformed and subtracted from the transformed attributes at the encoder side. The residual of alternating current (AC) coefficients will be signalled. Note that the prediction does not affect the direct current (DC) coefficient.

[0079] The each sub-node of transform node is predicted by 7 parent-level nodes where 3 coline parent-level neighbour nodes, 3 coplane parent-level neighbour nodes and 1 parent node. Coplane and coline neighbours are the neighbours that share a face and an edge with current transform node, respectively. A binary search algorithm is used to find coplane and coline parent-level neighbours. FIG. **4** illustrates parent-level nodes for each sub-node of transform unit node. FIG. **4** shows 7 parent-level nodes for each sub-node of transform node.

[0080] The attribute an, of each sub-node is predicted depending on the distance between it and its parent-level node as follows.

$$a_{up} = \sum \omega_k a_k / \sum \omega_k$$

Where a_k is the attribute of its one parent-level node and ω_k is weight depending on the distance. In G-PCC, ω_{parent}:$\omega_{coplane}$:ω_{coline}=4:2:1.

3.2.2 Early Termination for Transform Domain Prediction

[0081] Early termination is introduced to reduce complexity. In the upsampled transform domain prediction, 7 parent-level neighbour nodes are used to create the prediction value for each encoding target node (sub-node) of transform node. And there are total 19 parent-level neighbour nodes (containing parent node, i.e. transform unit node) which are used to create the prediction value for all 8 encoding target nodes of transform unit node. The prediction accuracy would be better in the denser point cloud because the number of valid neighbour parent nodes is larger. On the contrary, the prediction accuracy would be worse in the sparser point cloud. Based on this feature, the early termination for upsampled the transform domain prediction is introduced to reduce the coding time. In early termination, the following two parameters in every 8 sub-nodes of transform unit node are calculated.

[0082] NumValidP: total number of valid parent-level neighbour node (containing parent node).

[0083] NumValidGP: total number of valid grandparent-level neighbour node (containing grandparent node).

[0084] Then, the prediction will be disable in case that either NumValidP or NumValidGP is less than threshold. It means that the prediction is terminated when the number of valid neighbour nodes becomes small.

3.3 Problems

[0085] The existing designs for point cloud attribute transform domain prediction in region-adaptive hierarchical transform have the following problems:

[0086] 1. In current design, the binary search algorithm is used to find coplane and coline parent-level neighbours. However, the binary search is performed frequently in neighbor search process and causes a high time complexity. If the binary search algorithm can be replaced by a faster search algorithm, e.g., look-up table algorithm, a lot of coding time can be saved.

4. Detailed Solutions

[0087] To solve the above problems and some other problems not mentioned, methods as summarized below are disclosed. The embodiments should be considered as examples to explain the general concepts and should not be interpreted in a narrow way. Furthermore, these embodiments can be applied individually or combined in any manner.

[0088] 1) The node may be stored at decoder and/or encoder.

[0089] a. In one example, the node may be stored in a data structure.

[0090] b. In one example, the data structure may be array (may be 1D or n-D), list, map and so on.

[0091] c. In one example, the node index may be the unique indicator that refers to one node in the data structure.

[0092] i. In one example, the node index may be the position of one node in the data structure.

[0093] ii. In one example, the node index may be the pointer that points to one node in the data structure.

[0094] 2) A lookup table may be used to store at least one node index.

[0095] a. In one example, suppose the lookup table is T[f], then T[f] is a node index and f may be Morton codes, Hilbert codes or other converted codes of the node position.

[0096] i. In one example, the converted codes may be the converted codes of partial bits of the node position.

[0097] ii. In one example, the converted codes may be the converted codes of all bits of the node position.

[0098] b. In one example, suppose the lookup table is T[x][y][z], then T[x][y][z] is a node index and x, y, z may be the node position.

[0099] i. In one example, the point position may be the partial bits of the node position.

[0100] ii. In one example, the point position may be the all bits of the node position.

[0101] c. In one example, the lookup table may be 1D, 2D or 3D.

[0102] d. In one example, the lookup table may be a 3D array.

[0103] e. In one example, the lookup table may be a hash table.

[0104] f. In one example, the index of the lookup table may be Morton codes, Hilbert codes or other converted codes of the node position.

[0105] i. In one example, the converted codes may be the converted codes of partial bits of the node position.

[0106] ii. In one example, the converted codes may be the converted codes of all bits of the node position.

[0107] g. In one example, the index of the lookup table may be the node position.

[0108] i. In one example, the point position may be the partial bits of the node position.

[0109] ii. In one example, the point position may be the all bits of the node position.

[0110] h. In one example, an empty index may be used to indicate that one node does not exist.

[0111] i. In one example, the empty index may be one pre-defined index.

[0112] 1. In one example, the pre-defined index can only be used when the corresponding node does not exist.

[0113] 2. In one example, the pre-defined index may be -1.

[0114] 3) The node whose indices are stored in the lookup table may be in a limited region.

[0115] a. In one example, the nodes whose indices are stored in the lookup table may share the same octree depth.

[0116] b. In one example, the nodes whose indices are stored in the lookup table may have different octree depth.

[0117] c. In one example, the limited region may be a regular region.

[0118] i. In one example, the limited region may be a rectangular region.

[0119] ii. In one example, the limited region may be a square region.

[0120] 1. In one example, the square region may correspond to one octree node in one octree depth.

[0121] d. In one example, the limited region may be an irregular region.

[0122] i. In one example, the limited region may contain a rectangular region and its neighbour nodes.

[0123] 1. In one example, the neighbour nodes may be the nodes that shares at least a face, or an edge, or a vertex with the rectangular region.

[0124] ii. In one example, the limited region may contain a square region and its neighbour nodes.

[0125] 1. In one example, the neighbour nodes may be the nodes that shares at least a face, or an edge, or a vertex with the square region.

[0126] iii. In above description, the neighbours nodes may share a same octree depth with the node.

[0127] e. At least one indication may be used to indicate the size of the limited region.

[0128] i. In one example, the indication may be the size of the limited region.

[0129] ii. In one example, the size of the limited region may be a function of the indication value.

[0130] 1. In one example, the function may be linear function, power function, exponential function or logarithmic function and so on.

[0131] iii. In one example, one indication may indicate the size of one dimension of the limited region.

[0132] iv. The indication may be signaled to the decoder.

[0133] 1. In one example, the indication may be coded with fixed-length coding, unary coding, truncated unary coding and so on.

[0134] 2. In one example, the indication may be coded with at least one context in arithmetic coding.

[0135] 3. In one example, the indication may be bypass coded.

[0136] 4. In one example, the indication may be coded in predictive way.

[0137] 5. The indication may be a constant that is same in decoder side and encoder side.

[0138] 4) The lookup table may be used for neighbour search.

[0139] a. In one example, an empty index may be used to indicate that one node does not exist.

[0140] i. In one example, the empty index may be one pre-defined index.

[0141] 1. In one example, the pre-defined index can only be used when the corresponding node does not exist.

[0142] 2. In one example, the pre-defined index may be −1.

[0143] b. In one example, the lookup table may be initialized by the empty index.

[0144] c. In one example, the lookup table may be initialized before neighbour search and prediction process.

[0145] i. In one example, the lookup table may be initialized by visiting all the nodes in the limited region.

[0146] d. In one example, the lookup table may be reset as empty index after the prediction process.

[0147] i. In one example, the lookup table may be reset as empty index by visiting all the nodes in the limited region.

[0148] e. In one example, the neighbor search may be performed based on the lookup table.

[0149] i. In one example, if the index of one node is not the empty index in the lookup table, the node can be found by the index stored in the lookup table.

[0150] ii. In one example, if the index of one node is the empty index in the lookup table, the node can be derived as an empty node.

[0151] 5) One indication may be used to indicate the memory size of the lookup table.

[0152] a. In one example, the indication may be the memory size of the lookup table.

[0153] b. In one example, the memory size of the lookup table may be a function of the indication value.

[0154] i. In one example, the function may be linear function, power function, exponential function or logarithmic function and so on.

[0155] c. The indication may be signaled to the decoder.

[0156] i. In one example, the indication may be coded with fixed-length coding, unary coding, truncated unary coding and so on.

[0157] ii. In one example, the indication may be coded with at least one context in arithmetic coding.

[0158] iii. In one example, the indication may be bypass coded.

[0159] iv. In one example, the indication may be coded in predictive way.

[0160] d. The indication may be a constant that is same in decoder side and encoder side.

[0161] 6) One indication may be used to indicate whether the lookup table is used to store the node indexes.

[0162] a. The indication may be signaled to the decoder.

[0163] i. In one example, the indication may be coded with fixed-length coding, unary coding, truncated unary coding and so on.

[0164] ii. In one example, the indication may be coded with at least one context in arithmetic coding.

[0165] iii. In one example, the indication may be bypass coded.

[0166] iv. In one example, 19170020e indication may be coded in predictive way.

[0167] b. The indication may be a constant that is same in decoder side and encoder side.

[0168] 7) The lookup table may be built up at decoder and/or encoder.

[0169] a. In one example, if the lookup table is a 3D table and the node position is the index of the lookup table, i.e., T[x][y][z] is the index of the node whose position is (x,y,z), the memory may be assigned to the 3D lookup table and all the entries of the lookup table may be initialized by the empty index.

[0170] b. In one example, initialize the lookup table by visiting all the nodes in the limited region that it stores, i.e., set corresponding node index in the 3D lookup table for every node in the limited region.

[0171] 8) The lookup table may be built up depending on information signaled from encoder to decoder.

[0172] a. In one example, if the lookup table is a 3D table and the node position is the index of the lookup table, i.e., T[x][y][z] is the index of the node whose position is (x,y,z), the memory may be assigned to the 3D lookup table and all the entries of the lookup table may be initialized by the empty index.

[0173] i. In one example, the memory size of the lookup table may be determined by an indication and the indication may be signaled from encoder to decoder.

[0174] b. In one example, initialize the lookup table by visiting all the nodes in the limited region that it stores, i.e., set corresponding node index in the 3D lookup table for every node in the limited region.

5. Embodiments

[0175] FIG. **5** illustrates an example of the coding flow **500** in accordance with embodiments of the present disclosure. At **510**, a lookup table is constructed according to an indication determining the memory size of the lookup table. A**520**, the lookup table is initialized with an empty index. Then, at **530**, the lookup table is initialized by visiting all the nodes in the limited region that it stores. At **540**, a neighbour index is searched in the lookup table if necessary when coding the limited region, such as RAHT prediction. At **550**, it is determined whether the neighbour index is equal to the empty index.

[0176] If yes, the flow **500** goes to the "YES" branch and look for the neigbhor. At **560**, it may be determined that the neighbor does not exist and the flow **500** then goes to **580** at which the lookup table is reset with the empty index after coding the limited region.

[0177] On the other hand, if the neighbour index is determined to be not equal to the empty index at **550**, the flow **500** goes to the "NO" branch, and at **570**, the neighbour may be found and the information is to be used. At **580**, the lookup table is reset with the empty index after coding the limited region.

[0178] More details will be further discussed below. FIG. **6** illustrates a flowchart of a method **600** for point cloud coding in accordance with embodiments of the present disclosure.

[0179] It is to be understood that in the following embodiments, the term "block" refer to a space in which may have point(s) or may not have any point of the point cloud, and the point(s) in the block will be processed during the conversion between a point cloud sequence and its bitstream. A block may comprise one or more subblocks and each subblock may or may not include point(s). A block may have one or more neighbor blocks which is adjacent to the block.

[0180] As to the term "node", it refers to a node representing or corresponding to a transform block, for example, in a tree level. A node may have sub-node(s) corresponding to subblocks of a block. A node may also have a parent node, for example, in a tree level, which corresponds to a parent block comprising the block.

[0181] At block **610**, a node index of a first node of a plurality of nodes is determined for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with the plurality of nodes and a bitstream of the point cloud sequence. The first node is stored in a data structure and is indicated in the data structure by the node index.

[0182] At block **620**, the conversion is performed based on the position indication and the occupancy information.

[0183] The method **600** enables reducing of the time complexity of search. With the fast search solution, such as the look-up table algorithm, proposed by embodiments of the present disclosure, a lot of coding time can be saved.

[0184] In some embodiments, the first node may be stored at a decoder and/or an encoder side.

[0185] In some embodiments, the data structure further comprises occupancy information indicating whether there is a point in the first node.

[0186] In some embodiments, the data structure may be in the form of an array, a list, or a map.

[0187] In some embodiments, the node index may be a node position of the first node in the data structure. Alternatively, the node index may be a pointer that points to the first node in the data structure.

[0188] In some embodiments, the node index may be stored in a lookup table.

[0189] In some embodiments, in the lookup table, the node index may be denoted as T[f]. Alternatively, f indicates a converted code of a node position of the first node.

[0190] In some embodiments, the converted code may be a Morton code or a Hilbert code.

[0191] In some embodiments, the converted code may be a converted code of partial bits of the node position, or wherein the converted code may be a converted code of all bits of the node position.

[0192] In some embodiments, in the lookup table, the node index may be denoted as T[x][y][z], wherein x, y and z indicate a node position of the first node.

[0193] In some embodiments, a point position may be indicated by partial bits of the node position, or wherein the point position may be indicated by all bits of the node position.

[0194] In some embodiments, the lookup table may be 1D, 2D or 3D. Alternatively, the lookup table may be a 3D array. Alternatively, the lookup table may be a hash table.

[0195] In some embodiments, a node index in the lookup table may be a converted code of a node position of one of the plurality of nodes.

[0196] In some embodiments, the converted code may be a Morton code or a Hilbert code.

[0197] In some embodiments, the converted code may be a converted code of partial bits of the node position, or wherein the converted code may be a converted code of all bits of the node position.

[0198] In some embodiments, the converted code may be a converted code of partial bits of the node position, or wherein the converted code may be a converted code of all bits of the node position.

[0199] In some embodiments, a node index of the lookup table may be a node position.

[0200] In some embodiments, the point position may be indicated by partial bits of the node position, or all bits of the node position.

[0201] In some embodiments, an empty index may be used to indicate that one node does not exist.

[0202] In some embodiments, the empty index may be a pre-defined index.

[0203] In some embodiments, the pre-defined index can only be used when the corresponding node does not exist.

[0204] In some embodiments, the empty index may be **–1**.

[0205] In some embodiments, nodes whose indices are stored in the lookup table are in a limited region.

[0206] In some embodiments, the nodes whose indices are stored in the lookup table share the same tree level or have different tree levels.

[0207] In some embodiments, a tree level comprises an octree depth.

[0208] In some embodiments, the limited region may be a regular region.

[0209] In some embodiments, the limited region may be a rectangular region or a square region.

[0210] In some embodiments, the square region corresponds to one octree node in one octree depth.

[0211] In some embodiments, the limited region may be an irregular region.

[0212] In some embodiments, the limited region contains a rectangular region and its neighbour nodes. Alternatively, the limited region contains a square region and its neighbour nodes.

[0213] In some embodiments, the neighbour nodes are nodes that shares at least one of a face, or an edge, or a vertex with the rectangular region or the square region.

[0214] In some embodiments, the neighbours nodes share a same tree level with the first node.

[0215] In some embodiments, the tree level comprises an octree depth.

[0216] In some embodiments, at least one indication may be used to indicate a size of the limited region.

[0217] In some embodiments, the indication may be the size of the limited region, wherein the size of the limited region may be a function of a value of the indication, wherein the indication indicates a size of one dimension of the limited region. Alternatively, the indication may be indicated to a decoder.

[0218] In some embodiments, the function comprises at least one of a linear function, a power function, an exponential function or a logarithmic function.

[0219] In some embodiments, the indication may be coded with at least one of fixed-length coding, unary coding, or truncated unary coding. Alternatively, the indication may be coded with at least one context in arithmetic coding. Alternatively, the indication may be bypass coded. Alternatively, the indication may be coded in a predictive way. Alternatively, the indication may be a constant that is same at a decoder side and an encoder side.

[0220] In some embodiments, the lookup table may be used for neighbour search.

[0221] In some embodiments, an empty index may be used to indicate that one node does not exist.

[0222] In some embodiments, the empty index may be a pre-defined index.

[0223] In some embodiments, the pre-defined index is only used if a corresponding node does not exist, or wherein the pre-defined index is −1.

[0224] In some embodiments, the lookup table may be initialized by an empty index.

[0225] In some embodiments, the lookup table may be initialized before neighbour search and prediction process.

[0226] In some embodiments, the lookup table may be initialized by visiting all the nodes in the limited region.

[0227] In some embodiments, the lookup table may be reset as empty index after the prediction process.

[0228] In some embodiments, the lookup table may be reset as empty index by visiting all the nodes in the limited region.

[0229] In some embodiments, the neighbor search may be performed based on the lookup table.

[0230] In some embodiments, if an index of one node is not the empty index in the lookup table, the node can be found by the index stored in the lookup table. Alternatively, if an index of one node is the empty index in the lookup table, the node can be derived as an empty node.

[0231] In some embodiments, an indication may be used to indicate a memory size of the lookup table.

[0232] In some embodiments, the indication may be the memory size of the lookup table.

[0233] In some embodiments, the memory size of the lookup table may be a function of the indication value.

[0234] In some embodiments, the function may be one of a linear function, a power function, an exponential function or a logarithmic function.

[0235] In some embodiments, the indication may be indicated to a decoder.

[0236] In some embodiments, the indication may be coded with one of a fixed-length coding, a unary coding, or a truncated unary coding. Alternatively, the indication may be coded with at least one context in arithmetic coding. Alternatively, the indication may be bypass coded. Alternatively, the indication may be coded in a predictive way.

[0237] In some embodiments, the indication may be a constant that is same at a decoder side and an encoder side.

[0238] In some embodiments, an indication is used to indicate whether the lookup table is used to store the node index.

[0239] In some embodiments, the indication may be indicated to the decoder.

[0240] In some embodiments, the indication may be coded with one of a fixed-length coding, a unary coding, or a truncated unary coding. Alternatively, the indication may be coded with at least one context in arithmetic coding. Alternatively, the indication may be bypass coded. Alternatively, the indication may be coded in a predictive way.

[0241] In some embodiments, the indication may be a constant that is same at a decoder side and an encoder side.

[0242] In some embodiments, the lookup table may be built up at a decoder and/or an encoder.

[0243] In some embodiments, if the lookup table is a 3D table and a node position is the node index of the lookup table, a memory is assigned to the lookup table and each entry in the lookup table may be initialized by an empty index.

[0244] In some embodiments, the lookup table may be initialized by setting a corresponding node index in the lookup table for every node in the limited region.

[0245] In some embodiments, the lookup table may be built up depending on information indicated from an encoder to a decoder.

[0246] In some embodiments, if the lookup table is a 3D table and the node position is the node index of the lookup table, a memory is assigned to the lookup table and each entry in the lookup table is initialized by an empty index.

[0247] In some embodiments, a size of the memory of the lookup table may be determined by an indication, and the indication may be indicated from an encoder to a decoder.

[0248] In some embodiments, the lookup table may be initialized by setting a corresponding node index in the lookup table for every node in the limited region.

[0249] In some embodiments, the at least one PC sample comprises one of the following: a frame, a picture, a slice, a sub-frame, a sub-picture, a tile, or a segment.

[0250] In some embodiments, the conversion includes encoding the at least one PC sample into the bitstream.

[0251] In some embodiments, the conversion includes decoding the at least one PC sample from the bitstream.

[0252] According to further embodiments of the present disclosure, a non-transitory computer-readable recording medium is provided. The non-transitory computer-readable recording medium stores a bitstream of a point cloud sequence which is generated by a method performed by an apparatus for point cloud coding. The method comprises: determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; and generating the bitstream based on the position indication and the occupancy information.

[0253] According to still further embodiments of the present disclosure, a method for storing bitstream of a point cloud sequence is provided. The method comprises: determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; generating the bitstream based on the position indication and the occupancy information; and storing the bitstream in a non-transitory computer-readable recording medium.

[0254] Implementations of the present disclosure can be described in view of the following clauses, the features of which can be combined in any reasonable manner.

[0255] Clause 1. A method for point cloud coding, comprising: determining, for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with a plurality of nodes and a bitstream of the point cloud sequence, a node index of a first node of the plurality of nodes, wherein the first node is stored in a data structure and is indicated in the data structure by the node index; and performing the conversion based on the position indication and the occupancy information.

[0256] Clause 2. The method of clause 1, wherein the first node is stored at a decoder and/or an encoder side.

[0257] Clause 3. The method of clause 1, wherein the data structure further comprises occupancy information indicating whether there is a point in the first node.

[0258] Clause 4. The method of clause 1, wherein the data structure is in the form of an array, a list, or a map.

[0259] Clause 5. The method of clause 1, wherein the node index is a node position of the first node in the data structure, or wherein the node index is a pointer that points to the first node in the data structure.

[0260] Clause 6. The method of clause 1, wherein the node index is stored in a lookup table.

[0261] Clause 7. The method of clause 6, wherein in the lookup table, the node index is denoted as T[f], wherein f indicates a converted code of a node position of the first node.

[0262] Clause 8. The method of clause 7, wherein the converted code is a Morton code or a Hilbert code.

[0263] Clause 9. The method of clause 7 or 8, wherein the converted code is a converted code of partial bits of the node position, or wherein the converted code is a converted code of all bits of the node position.

[0264] Clause 10. The method of clause 6, wherein in the lookup table, the node index is denoted as T[x][y][z], wherein x, y and z indicate a node position of the first node.

[0265] Clause 11. The method of clause 10, wherein a point position is indicated by partial bits of the node position, or wherein the point position is indicated by all bits of the node position.

[0266] Clause 12. The method of clause 6, wherein the lookup table is 1D, 2D or 3D, or wherein the lookup table is a 3D array, or wherein the lookup table is a hash table.

[0267] Clause 13. The method of clause 6, wherein a node index in the lookup table is a converted code of a node position of one of the plurality of nodes.

[0268] Clause 14. The method of clause 13, wherein the converted code is a Morton code or a Hilbert code.

[0269] Clause 15. The method of clause 13 or 14, wherein the converted code is a converted code of partial bits of the node position, or wherein the converted code is a converted code of all bits of the node position.

[0270] Clause 16. The method of clause 13 or 14, wherein the converted code is a converted code of partial bits of the node position, or wherein the converted code is a converted code of all bits of the node position.

[0271] Clause 17. The method of clause 6, wherein a node index of the lookup table is a node position.

[0272] Clause 18. The method of clause 17, wherein the point position is indicated by partial bits of the node position, or all bits of the node position.

[0273] Clause 19. The method of clause 6, wherein an empty index is used to indicate that one node does not exist.

[0274] Clause 20. The method of clause 19, wherein the empty index is a pre-defined index.

[0275] Clause 21. The method of clause 20, wherein the pre-defined index can only be used when the corresponding node does not exist.

[0276] Clause 22. The method of clause 20, wherein the empty index is −1.

[0277] Clause 23. The method of clause 6, wherein nodes whose indices are stored in the lookup table are in a limited region.

[0278] Clause 24. The method of clause 23, wherein the nodes whose indices are stored in the lookup table share the same tree level or have different tree levels.

[0279] Clause 25. The method of clause 24, wherein a tree level comprises an octree depth.

[0280] Clause 26. The method of clause 23, wherein the limited region is a regular region.

[0281] Clause 27. The method of clause 23 or 26, wherein the limited region is a rectangular region or a square region.

[0282] Clause 28. The method of clause 27, wherein the square region corresponds to one octree node in one octree depth.

[0283] Clause 29. The method of clause 23, wherein the limited region is an irregular region.

[0284] Clause 30. The method of clause 29, wherein the limited region contains a rectangular region and its neighbour nodes, or wherein the limited region contains a square region and its neighbour nodes.

[0285] Clause 31. The method of clause 30, wherein the neighbour nodes are nodes that shares at least one of a face, or an edge, or a vertex with the rectangular region or the square region.

[0286] Clause 32. The method of any of clauses 29 to 31, wherein the neighbours nodes share a same octree depth with the first node.

[0287] Clause 33. The method of clause 23, wherein at least one indication is used to indicate a size of the limited region.

[0288] Clause 34. The method of clause 33, wherein the indication is the size of the limited region, wherein the size of the limited region is a function of a value of the indication, wherein the indication indicates a size of one dimension of the limited region, or wherein the indication is indicated to a decoder.

[0289] Clause 35. The method of clause 34, wherein the function comprises at least one of a linear function, a power function, an exponential function or a logarithmic function.

[0290] Clause 36. The method of clause 34, wherein the indication is coded with at least one of fixed-length coding, unary coding, or truncated unary coding, or wherein the indication is coded with at least one context in arithmetic coding, or wherein the indication is bypass coded, or wherein the indication is coded in a predictive way, or wherein the indication is a constant that is same at a decoder side and an encoder side.

[0291] Clause 37. The method of clause 6, wherein the lookup table is used for neighbour search.

[0292] Clause 38. The method of clause 37, wherein an empty index is used to indicate that one node does not exist.

[0293] Clause 39. The method of clause 38, wherein the empty index is a pre-defined index.

[0294] Clause 40. The method of clause 39, wherein the pre-defined index is only be used if a corresponding node does not exist, or wherein the pre-defined index is −1.

[0295] Clause 41. The method of clause 37, wherein the lookup table is initialized by an empty index.

[0296] Clause 42. The method of clause 37, wherein the lookup table is initialized before neighbour search and prediction process.

[0297] Clause 43. The method of clause 42, wherein the lookup table is initialized by visiting all the nodes in the limited region.

[0298] Clause 44. The method of clause 37, wherein the lookup table is reset as empty index after the prediction process.

[0299] Clause 45. The method of clause 44, wherein the lookup table is reset as empty index by visiting all the nodes in the limited region.

[0300] Clause 46. The method of clause 37, wherein the neighbor search is performed based on the lookup table.

[0301] Clause 47. The method of clause 46, wherein if an index of one node is not the empty index in the lookup table, the node can be found by the index stored in the lookup table, or wherein if an index of one node is the empty index in the lookup table, the node can be derived as an empty node.

[0302] Clause 48. The method of clause 6, wherein an indication is used to indicate a memory size of the lookup table.

[0303] Clause 49. The method of clause 48, wherein the indication is the memory size of the lookup table.

[0304] Clause 50. The method of clause 48, wherein the memory size of the lookup table is a function of the indication value.

[0305] Clause 51. The method of clause 50, wherein the function is one of a linear function, a power function, an exponential function or a logarithmic function.

[0306] Clause 52. The method of clause 48, wherein the indication is indicated to a decoder.

[0307] Clause 53. The method of clause 52, wherein the indication is coded with one of a fixed-length coding, a unary coding, or a truncated unary coding, or wherein the indication is coded with at least one context in arithmetic coding, or wherein the indication is bypass coded, or wherein the indication is coded in a predictive way.

[0308] Clause 54. The method of clause 48, wherein the indication is a constant that is same at a decoder side and an encoder side.

[0309] Clause 55. The method of clause 6, wherein an indication is used to indicate whether the lookup table is used to store the node index.

[0310] Clause 56. The method of clause 55, wherein the indication is indicated to the decoder.

[0311] Clause 57. The method of clause 56, wherein the indication is coded with one of a fixed-length coding, a unary coding, or a truncated unary coding, or wherein the indication is coded with at least one context in arithmetic coding, or wherein the indication is bypass coded, or wherein the indication is coded in a predictive way.

[0312] Clause 58. The method of clause 55, wherein the indication is a constant that is same at a decoder side and an encoder side.

[0313] Clause 59. The method of clause 6, wherein the lookup table is built up at a decoder and/or an encoder.

[0314] Clause 60. The method of clause 59, wherein if the lookup table is a 3D table and a node position is the node index of the lookup table, a memory is assigned to the lookup table and each entry in the lookup table is initialized by an empty index.

[0315] Clause 61. The method of clause 59, wherein the lookup table is initialized by setting a corresponding node index in the lookup table for every node in the limited region.

[0316] Clause 62. The method of clause 6, wherein the lookup table is built up depending on information indicated from an encoder to a decoder.

[0317] Clause 63. The method of clause 62, wherein if the lookup table is a 3D table and the node position is the node index of the lookup table, a memory is assigned to the lookup table and each entry in the lookup table is initialized by an empty index.

[0318] Clause 64. The method of clause 63, wherein a size of the memory of the lookup table is determined by an indication, and the indication is indicated from an encoder to a decoder.

[0319] Clause 65. The method of clause 63, wherein the lookup table is initialized by setting a corresponding node index in the lookup table for every node in the limited region.

[0320] Clause 66. The method of any of clauses 1-65, wherein the at least one PC sample comprises one of the following: a frame, a picture, a slice, a sub-frame, a sub-picture, a tile, or a segment.

[0321] Clause 67. The method of any of clauses 1-66, wherein the conversion includes encoding the at least one PC sample into the bitstream.

[0322] Clause 68. The method of any of clauses 1-66, wherein the conversion includes decoding the at least one PC sample from the bitstream.

[0323] Clause 69. An apparatus for point cloud coding comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform a method in accordance with any of clauses 1-68.

[0324] Clause 70. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method in accordance with any of clauses 1-68.

[0325] Clause 71. A non-transitory computer-readable recording medium storing a bitstream of a point cloud sequence which is generated by a method performed by an apparatus for point cloud coding, wherein the method comprises: determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; and generating the bitstream based on the position indication and the occupancy information.

[0326] Clause 72. A method for storing a bitstream of a point cloud sequence, comprising: determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; generating the bitstream based on the position indication and the occupancy information; and storing the bitstream in a non-transitory computer-readable recording medium.

Example Device

[0327] FIG. **7** illustrates a block diagram of a computing device **700** in which various embodiments of the present disclosure can be implemented. The computing device **700** may be implemented as or included in the source device **110** (or the GPCC encoder **116** or **200**) or the destination device **120** (or the GPCC decoder **126** or **300**).

[0328] It would be appreciated that the computing device **700** shown in FIG. **7** is merely for purpose of illustration, without suggesting any limitation to the functions and scopes of the embodiments of the present disclosure in any manner.

[0329] As shown in FIG. **7**, the computing device **700** includes a general-purpose computing device **700**. The computing device **700** may at least comprise one or more processors or processing units **710**, a memory **720**, a storage unit **730**, one or more communication units **740**, one or more input devices **750**, and one or more output devices **760**.

[0330] In some embodiments, the computing device **700** may be implemented as any user terminal or server terminal having the computing capability. The server terminal may be a server, a large-scale computing device or the like that is provided by a service provider. The user terminal may for example be any type of mobile terminal, fixed terminal, or portable terminal, including a mobile phone, station, unit, device, multimedia computer, multimedia tablet, Internet node, communicator, desktop computer, laptop computer, notebook computer, netbook computer, tablet computer, personal communication system (PCS) device, personal navigation device, personal digital assistant (PDA), audio/video player, digital camera/video camera, positioning device, television receiver, radio broadcast receiver, E-book device, gaming device, or any combination thereof, including the accessories and peripherals of these devices, or any combination thereof. It would be contemplated that the computing device **700** can support any type of interface to a user (such as "wearable" circuitry and the like).

[0331] The processing unit **710** may be a physical or virtual processor and can implement various processes based on programs stored in the memory **720**. In a multi-processor system, multiple processing units execute computer executable instructions in parallel so as to improve the parallel processing capability of the computing device **700**. The processing unit **710** may also be referred to as a central processing unit (CPU), a microprocessor, a controller or a microcontroller.

[0332] The computing device **700** typically includes various computer storage medium. Such medium can be any medium accessible by the computing device **700**, including, but not limited to, volatile and non-volatile medium, or detachable and non-detachable medium. The memory **720** can be a volatile memory (for example, a register, cache, Random Access Memory (RAM)), a non-volatile memory (such as a Read-Only Memory (ROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), or a flash memory), or any combination thereof. The storage unit **730** may be any detachable or non-detachable medium and may include a machine-readable medium such as a memory, flash memory drive, magnetic disk or another other media, which can be used for storing information and/or data and can be accessed in the computing device **700**.

[0333] The computing device **700** may further include additional detachable/non-detachable, volatile/non-volatile memory medium. Although not shown in FIG. **7**, it is possible to provide a magnetic disk drive for reading from and/or writing into a detachable and non-volatile magnetic disk and an optical disk drive for reading from and/or writing into a detachable non-volatile optical disk. In such cases, each drive may be connected to a bus (not shown) via one or more data medium interfaces.

[0334] The communication unit **740** communicates with a further computing device via the communication medium. In addition, the functions of the components in the computing device **700** can be implemented by a single computing cluster or multiple computing machines that can communicate via communication connections. Therefore, the computing device **700** can operate in a networked environment using a logical connection with one or more other servers, networked personal computers (PCs) or further general network nodes.

[0335] The input device **750** may be one or more of a variety of input devices, such as a mouse, keyboard, tracking ball, voice-input device, and the like. The output device **760** may be one or more of a variety of output devices, such as a display, loudspeaker, printer, and the like. By means of the communication unit **740**, the computing device **700** can further communicate with one or more external devices (not shown) such as the storage devices and display device, with one or more devices enabling the user to interact with the computing device **700**, or any devices (such as a network card, a modem and the like) enabling the computing device **700** to communicate with one or more other computing devices, if required. Such communication can be performed via input/output (I/O) interfaces (not shown).

[0336] In some embodiments, instead of being integrated in a single device, some or all components of the computing device **700** may also be arranged in cloud computing architecture. In the cloud computing architecture, the components may be provided remotely and work together to implement the functionalities described in the present disclosure. In some embodiments, cloud computing provides computing, software, data access and storage service, which will not require end users to be aware of the physical locations or configurations of the systems or hardware providing these services. In various embodiments, the cloud computing provides the services via a wide area network (such as Internet) using suitable protocols. For example, a cloud computing provider provides applications over the wide area network, which can be accessed through a web browser or any other computing components. The software or components of the cloud computing architecture and corresponding data may be stored on a server at a remote position. The computing resources in the cloud computing environment may be merged or distributed at locations in a remote data center. Cloud computing infrastructures may provide the services through a shared data center, though they behave as a single access point for the users. Therefore, the cloud computing architectures may be used to provide the components and functionalities described herein from a service provider at a remote location. Alternatively, they may be provided from a conventional server or installed directly or otherwise on a client device.

[0337] The computing device **700** may be used to implement point cloud encoding/decoding in embodiments of the present disclosure. The memory **720** may include one or more point cloud coding modules **725** having one or more program instructions. These modules are accessible and executable by the processing unit **710** to perform the functionalities of the various embodiments described herein.

[0338] In the example embodiments of performing point cloud encoding, the input device **750** may receive point cloud data as an input **770** to be encoded. The point cloud data may be processed, for example, by the point cloud coding module **725**, to generate an encoded bitstream. The encoded bitstream may be provided via the output device **760** as an output **780**.

[0339] In the example embodiments of performing point cloud decoding, the input device **750** may receive an encoded bitstream as the input **770**. The encoded bitstream may be processed, for example, by the point cloud coding module **725**, to generate decoded point cloud data. The decoded point cloud data may be provided via the output device **760** as the output **780**.

[0340] While this disclosure has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the present application as defined by the appended claims. Such variations are intended to be covered by the scope of this present application. As such, the foregoing description of embodiments of the present application is not intended to be limiting.

1. A method for point cloud coding, comprising:

determining, for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with a plurality of nodes and a bitstream of the point cloud sequence, a node index of a first node of the plurality of nodes, wherein the first node is stored in a data structure and is indicated in the data structure by the node index; and

performing the conversion based on a position indication and occupancy information.

2. The method of claim **1**, wherein the first node is stored at a decoder and/or an encoder side, and/or

wherein the data structure further comprises the occupancy information indicating whether there is a point in the first node, and/or

wherein the data structure is in the form of an array, a list, or a map, and/or

wherein the node index is a node position of the first node in the data structure, or wherein the node index is a pointer that points to the first node in the data structure, and/or

wherein the at least one PC sample comprises one of the following: a frame, a picture, a slice, a sub-frame, a sub-picture, a tile, or a segment.

3. The method of claim **1**, wherein the node index is stored in a lookup table.

4. The method of claim **3**, wherein in the lookup table, the node index is denoted as T[f], wherein f indicates a converted code of a node position of the first node, wherein the converted code is one of: a Morton code, a Hilbert code, a converted code of partial bits of the node position, or a converted code of all bits of the node position, or

wherein in the lookup table, the node index is denoted as T[x][y][z], wherein x, y and z indicate a node position of the first node, wherein a point position is indicated by partial bits of the node position, or wherein the point position is indicated by all bits of the node position, and/or

wherein the lookup table is one of: 1D, 2D, 3D, a 3D array, or a hash table.

5. The method of claim **3**, wherein a node index in the lookup table is a converted code of a node position of one of the plurality of nodes, wherein the converted code is a Morton code or a Hilbert code,

wherein the converted code is a converted code of partial bits of the node position, or

wherein the converted code is a converted code of all bits of the node position.

6. The method of claim **3**, wherein a node index of the lookup table is a node position, wherein the point position is indicated by partial bits of the node position, or all bits of the node position, and/or

wherein an empty index is used to indicate that one node does not exist, wherein the empty index is a pre-defined index, wherein the pre-defined index can only be used when the corresponding node does not exist, and/or wherein the empty index is −1.

7. The method of claim **3**, wherein nodes whose indices are stored in the lookup table are in a limited region, wherein the nodes whose indices are stored in the lookup table share the same tree level or have different tree levels, wherein a tree level comprises an octree depth.

8. The method of claim **7**, wherein the limited region is a regular region, wherein the limited region is a rectangular region or a square region, wherein the square region corresponds to one octree node in one octree depth.

9. The method of claim **7**, wherein the limited region is an irregular region, wherein the limited region contains a rectangular region and its neighbour nodes, or wherein the limited region contains a square region and its neighbour nodes,

wherein the neighbour nodes are nodes that shares at least one of a face, or an edge, or a vertex with the rectangular region or the square region, and/or

wherein the neighbours nodes share a same octree depth with the first node.

10. The method of claim **7**, wherein at least one indication is used to indicate a size of the limited region,

wherein the indication is the size of the limited region,

wherein the size of the limited region is a function of a value of the indication,

wherein the indication indicates a size of one dimension of the limited region, or

wherein the indication is indicated to a decoder.

11. The method of claim **10**, wherein the function comprises at least one of a linear function, a power function, an exponential function or a logarithmic function, and/or

wherein the indication is coded with at least one of fixed-length coding, unary coding, or truncated unary coding, or

wherein the indication is coded with at least one context in arithmetic coding, or

wherein the indication is bypass coded, or
wherein the indication is coded in a predictive way, or
wherein the indication is a constant that is same at a decoder side and an encoder side.

12. The method of claim **3**, wherein the lookup table is used for neighbour search, wherein an empty index is used to indicate that one node does not exist, wherein the empty index is a pre-defined index,
wherein the pre-defined index is only used if a corresponding node does not exist, or
wherein the pre-defined index is −1.

13. The method of claim **3**, wherein the lookup table is used for neighbour search,
wherein the lookup table is initialized by an empty index, wherein the lookup table is initialized before neighbour search and prediction process, and/or wherein the lookup table is initialized by visiting all the nodes in a limited region, or
wherein the lookup table is reset as empty index after the prediction process, wherein the lookup table is reset as empty index by visiting all the nodes in the limited region, or
wherein the neighbor search is performed based on the lookup table, wherein if an index of one node is not the empty index in the lookup table, the node can be found by the index stored in the lookup table, or wherein if an index of one node is the empty index in the lookup table, the node can be derived as an empty node.

14. The method of claim **3**, wherein an indication is used to indicate a memory size of the lookup table,
wherein the indication is the memory size of the lookup table, or
wherein the memory size of the lookup table is a function of the indication value, wherein the function is one of a linear function, a power function, an exponential function or a logarithmic function, or
wherein the indication is indicated to a decoder, wherein the indication is coded with one of a fixed-length coding, a unary coding, or a truncated unary coding, or wherein the indication is coded with at least one context in arithmetic coding, or wherein the indication is bypass coded, or wherein the indication is coded in a predictive way, or
wherein the indication is a constant that is same at a decoder side and an encoder side.

15. The method of claim **3**, wherein an indication is used to indicate whether the lookup table is used to store the node index, or
wherein the indication is indicated to the decoder, wherein the indication is coded with one of a fixed-length coding, a unary coding, or a truncated unary coding, or wherein the indication is coded with at least one context in arithmetic coding, or wherein the indication is bypass coded, or wherein the indication is coded in a predictive way, or
wherein the indication is a constant that is same at a decoder side and an encoder side.

16. The method of claim **3**, wherein the lookup table is built up at a decoder and/or an encoder, wherein if the lookup table is a 3D table and a node position is the node index of the lookup table, a memory is assigned to the lookup table and each entry in the lookup table is initialized by an empty index, or wherein the lookup table is initialized by setting a corresponding node index in the lookup table for every node in a limited region, or
wherein the lookup table is built up depending on information indicated from an encoder to a decoder, wherein if the lookup table is a 3D table and the node position is the node index of the lookup table, a memory is assigned to the lookup table and each entry in the lookup table is initialized by an empty index, wherein a size of the memory of the lookup table is determined by an indication, and the indication is indicated from an encoder to a decoder, or wherein the lookup table is initialized by setting a corresponding node index in the lookup table for every node in the limited region.

17. The method of claim **1**, wherein the conversion includes encoding the at least one PC sample into the bitstream, or
wherein the conversion includes decoding the at least one PC sample from the bitstream.

18. An apparatus for point cloud coding comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform a method comprising:
determining, for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with a plurality of nodes and a bitstream of the point cloud sequence, a node index of a first node of the plurality of nodes, wherein the first node is stored in a data structure and is indicated in the data structure by the node index; and
performing the conversion based on a position indication and occupancy information.

19. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method comprising:
determining, for a conversion between a point cloud sequence comprising at least one point cloud (PC) sample associated with a plurality of nodes and a bitstream of the point cloud sequence, a node index of a first node of the plurality of nodes, wherein the first node is stored in a data structure and is indicated in the data structure by the node index; and
performing the conversion based on a position indication and occupancy information.

20. A non-transitory computer-readable recording medium storing a bitstream of a point cloud sequence which is generated by a method performed by an apparatus for point cloud coding, wherein the method comprises:
determining a node index of a first node of a plurality of nodes, wherein the point cloud sequence comprises at least one point cloud (PC) sample associated with the plurality of nodes, and the first node is stored in a data structure and is indicated in the data structure by the node index; and
generating the bitstream based on a position indication and occupancy information.

* * * * *