

METHOD AND APPARATUS OF ENTROPY ENCODING/DECODING POINT CLOUD GEOMETRY DATA CAPTURED BY A SPINNING SENSORS HEAD

CROSS-REFERENCE TO RELATED APPLICATION

- 5 This application claims priority to European Patent Application No. EP20306673.3 filed on December 23, 2020, the content of which is hereby incorporated by reference in its entirety into this disclosure.

FIELD

- 10 The present application generally relates to point cloud compression and, in particular to methods and apparatus of entropy encoding/decoding point cloud geometry data captured by a spinning sensor head.

BACKGROUND

- 15 The present section is intended to introduce the reader to various aspects of art, which may be related to various aspects of at least one exemplary embodiments of the present application that is described and/or claimed below. This discussion is believed to be helpful in providing the reader with background information to facilitate a better understanding of the various aspects of the present application.

- 20 As a format for the representation of 3D data, point clouds have recently gained traction as they are versatile in their capability in representing all types of physical objects or scenes. Point clouds may be used for various purposes such as culture heritage/buildings in which objects like statues or buildings are scanned in 3D in order to share the spatial configuration of the object without sending or visiting it. Also, it is a way to ensure
25 preserving the knowledge of the object in case it may be destroyed; for instance, a temple by an earthquake. Such point clouds are typically static, colored and huge.

- Another use case is in topography and cartography in which using 3D representations allows for maps that are not limited to the plane and may include the relief. Google Maps is now a good example of 3D maps but uses meshes instead of point clouds. Nevertheless,
30 point clouds may be a suitable data format for 3D maps and such point clouds are typically static, colored and huge.

- Virtual Reality (VR), Augmented Reality (AR) and immersive worlds have recently become a hot topic and are foreseen by many as the future of 2D flat video. The basic idea is to immerse the viewer in a surrounding environment, in contrast to a standard TV that only
35 allows the viewer to look at the virtual world in front of him/her. There are several gradations in the immersivity depending on the freedom of the viewer in the environment. A point cloud is a good format candidate for distributing VR/AR worlds.

The automotive industry, and more particularly foreseen autonomous cars, are also domains in which point clouds may be intensively used. Autonomous cars should be able

to “probe” their environment to make good driving decisions based on the detected presence and nature of their immediate nearby objects and road configuration.

5 A point cloud is a set of points located in a tridimensional (3D) space, optionally with additional values attached to each of the points. These additional values are usually called attributes. Attributes may be, for example, three-component colors, material properties like reflectance and/or two-component normal vectors to a surface associated with a point.

A point cloud is thus a combination of a geometry (locations of the points in a 3D space usually represented by 3D cartesian coordinates x, y and z) and attributes.

10 Point clouds may be captured by various types of devices like an array of cameras, depth sensors, lasers (light detection and ranging, also known as Lidars), radars, or may be computer-generated (for example in movie post-production). Depending on the use cases, points clouds may have from thousands to up to billions of points for cartography applications. Raw representations of point clouds require a very high number of bits per point, with at least a dozen of bits per cartesian coordinate x, y or z , and optionally more bits for the attribute(s), for instance three times 10 bits for the colors.

15 It is important in many applications to be able to either distribute point clouds to an end-user or store them in a server by consuming only a reasonable amount of bit-rate or storage space, while maintaining an acceptable (or preferably very good) quality of experience. Efficient compression of these point clouds is a key point in order to make the distribution chain of many immersive worlds practical.

20 Compression may be lossy (like in video compression) for the distribution to and visualization by an end-user, for example on AR/VR glasses or any other 3D-capable device. Other use cases do require lossless compression, like medical applications or autonomous driving, to avoid altering the results of a decision obtained from the subsequent analysis of the compressed and transmitted point cloud.

25 Until recently, point cloud compression (aka PCC) was not addressed by the mass market and no standardized point cloud codec was available. In 2017, the standardization working group ISO/JCT1/SC29/WG11, also known as Moving Picture Experts Group or MPEG, has initiated work items on point cloud compression. This has led to two standards, namely

- MPEG-I part 5 (ISO/IEC 23090-5) or Video-based Point Cloud Compression (V-PCC)
- MPEG-I part 9 (ISO/IEC 23090-9) or Geometry-based Point Cloud Compression (G-PCC)

30 The V-PCC coding method compresses a point cloud by performing multiple projections of a 3D object to obtain 2D patches that are packed into an image (or a video when dealing with dynamic point clouds). Obtained images or videos are then compressed using already existing image/video codecs, allowing for the leverage of already deployed image and video solutions. By its very nature, V-PCC is efficient only on dense and continuous point clouds because image/video codecs are unable to compress non-smooth patches

40

as would be obtained from the projection of, for example, Lidar-captured sparse geometry data.

The G-PCC coding method has two schemes for the compression of a captured sparse geometry data.

- 5 The first scheme is based on an occupancy tree, being locally any type of tree among octree, quadtree or binary tree, representing the point cloud geometry. Occupied nodes are split down until a certain size is reached, and occupied leaf nodes provide the 3D locations of points, typically at the center of these nodes. The occupancy information is carried by occupancy flags signaling the occupancy status of each of the child nodes of nodes. By using neighbor-based prediction techniques, high level of compression of the occupancy flags can be obtained for dense point clouds. Sparse point clouds are also addressed by directly coding the position of point within a node with non-minimal size, by stopping the tree construction when only isolated points are present in a node; this technique is known as Direct Coding Mode (DCM).
- 10
- 15 The second scheme is based on a predictive tree in which each node represents the 3D location of one point and the parent/child relation between nodes represents spatial prediction from parent to children. **Figure 1** illustrates a portion of a predictive tree in which points (nodes) are represented by black circles and relationships parent and children points are represented by arrows. One child point has a unique parent point. Thus, a current point P_{n-1} has a unique parent point, a unique grandparent point and a unique great-grandparent point. Spatial predictors for the current point are constructed using these ancestor points, for example, the predictor may be the parent point itself, a linear prediction from grandparent to parent as shown on **Figure 2** or a parallelogram prediction from the three ancestors as shown on **Figure 3**. Then, a residual is constructed by subtracting the predictor to the current point. This residual is coded in the bitstream by using classical binarization and entropy coding techniques. This method can only address sparse point clouds and offers the advantage of lower latency and simpler decoding than the occupancy tree. However, compression performance is only marginally better, and the encoding is complex, relatively to the first occupancy-based method, because the encoder must intensively look for the best predictor (among a long list of potential predictors) when constructing the predictive tree.
- 20
- 25
- 30

35 In both schemes, attribute (de)coding is performed after complete geometry (de)coding, leading practically to a two-pass coding. Thus, the joint geometry/attribute low latency is obtained by using slices that decompose the 3D space into sub-volumes that are coded independently, without prediction between the sub-volumes. This may heavily impact the compression performance when many slices are used.

Combining together requirements on encoder and decoder simplicity, on low latency and on compression performance is still a problem that has not been satisfactory solved by existing point cloud codecs.

- 40 An important use case is the transmission of sparse geometry data captured by a spinning Lidar mounted on a moving vehicle. This usually requires a simple and low-latency embarked encoder. Simplicity is required because the encoder is likely to be deployed on

computing units which perform other processing in parallel, such as (semi-) autonomous driving, thus limiting the processing power available to the point cloud encoder. Low latency is also required to allow for fast transmission from the car to a cloud in order to have a real-time view of the local traffic, based on multiple-vehicle acquisition, and take
 5 adequate fast decision based on the traffic information. While transmission latency can be low enough by using 5G, the encoder itself shall not introduce too much latency due to coding. Also, compression performance is extremely important since the flow of data from millions of cars to the cloud is expected to be extremely heavy.

Specific priors related to spinning Lidar-captured sparse geometry data have been
 10 already exploited in G-PCC and have led to very significant gains of compression.

First, G-PCC exploits the elevation angle (relative to the horizontal ground) of capture from a spinning Lidar head as depicted on **Figures 4 and 5**. A Lidar head 10 comprises a set of sensors 11 (lasers), here five lasers are represented. The Lidar head 10 may spin around a vertical axis $\mathbf{z}$ to capture geometry data of a physical object. Lidar-captured
 15 geometry data is then represented in spherical coordinates (r_{3D}, ϕ, θ) , where r_{3D} is the distance of a point P from the Lidar head's center, ϕ is an azimuthal angle of the Lidar head's spin relative to a referential, and θ is an elevation angle of a sensor $\mathbf{k}$ of the Lidar head relative to a horizontal referential plane.

A regular distribution along the azimuthal angle has been observed on Lidar captured data as depicted on **Figure 6**. This regularity is used in G-PCC to obtain a quasi 1D representation of the point cloud where, up to noise, only a radius r_{3D} belongs to a continuous range of values while the angles ϕ and θ take only a discrete number of values $\phi_i \forall i = 0 \text{ to } I - 1$ where I is a number of azimuthal angles used for the capture of the points and $\theta_k \forall k = 0 \text{ to } K - 1$ where K is a number of sensors of the Lidar head 10.
 25 Basically, G-PCC represents Lidar-captured sparse geometry data on a 2D discrete angular plane (ϕ, θ) as depicted on **Figure 6**, together with a radius value r_{3D} for each point.

This quasi 1D property has been exploited in G-PCC in both the occupancy tree and the predictive tree by predicting, in the spherical coordinates space, the location of a current point based on an already coded point by using the discrete nature of angles.
 30

More precisely, the occupancy tree uses DCM intensively and entropy codes the direct locations of points within a node by using a context-adaptive entropy coder. Contexts are then obtained from the local conversion of the point locations into angular coordinates (ϕ, θ) and from the location of these angular coordinates relative to discrete angular coordinates (ϕ_i, θ_k) obtained from precedingly coded points. The predictive tree directly codes a first version of a point location in the angular coordinates (r_{2D}, ϕ, θ) , where r_{2D} is the projected radius on the horizontal xy plane as depicted on **Figure 7**, using the quasi 1D nature $(r_{2D}, \phi_i, \theta_k)$ of this coordinate space. Then, angular coordinates (r_{2D}, ϕ, θ) are converted into 3D cartesian coordinates (x, y, z) and a xyz residual is coded to tackle the errors of coordinate conversion, the approximation of elevation and azimuthal angles and potential noise.
 40

elevation and azimuthal angles and potential noise.

G-PCC does use the angular priors to better compress spinning Lidar-captured sparse geometry data but does not adapt the coding structure to the order of capture. By its very nature, the occupancy tree must be coded down to its last depth before outputting a point. This occupancy is coded in the so-called breadth-first order: the occupancy of the root node is first coded, indicating its occupied child nodes; then the occupancy for each of the occupied child nodes is coded, indicating the occupied grand-child nodes; and so on iteratively over the tree depth until leaf nodes can be determined and the corresponding points are provided/output to an application or to the attribute(s) coding scheme. Regarding the predictive tree, the encoder is free to choose the order of point in the tree, but to obtain good compression performance, to optimize the prediction accuracy, G-PCC proposes to code one tree per laser, as depicted on **Figure 8**. This has mainly the same drawback as using one coding slice per laser, i.e. non-optimal compression performance because prediction between lasers (sensors) is not allowed and does not provide encoder low latency. Worse, one should have one coding processing per laser (sensor) and the number of core coding units should equal the number of sensing lasers; this is not practical.

In brief, in a framework of a spinning sensors head used for capturing sparse geometry data of a point cloud, prior arts do not solve the problem of combining encoding and decoding simplicity, low latency and compression performance.

At least one exemplary embodiment of the present application has been devised with the foregoing in mind.

SUMMARY

The following section presents a simplified summary of at least one exemplary embodiment in order to provide a basic understanding of some aspects of the present application. This summary is not an extensive overview of an exemplary embodiment. It is not intended to identify key or critical elements of an embodiment. The following summary merely presents some aspects of at least one of the exemplary embodiments in a simplified form as a prelude to the more detailed description provided elsewhere in the document.

According to a first aspect of the present application, there is provided a method of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object, points of the point cloud being ordered based on azimuthal angles representing capture angles of sensors and sensor indices associated with sensors. The method comprises encoding, into a bitstream, at least one order index difference representing a difference between order indices of two consecutive ordered points by: obtaining at least one binary data by binarizing the at least one order index difference; and for each binary data, selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already encoded point, and context-based entropy coding the binary data in the bitstream, based on the selected context.

According to an exemplary embodiment, the already encoded point is a penultimate already coded point with a same sensor index as a sensor index associated with the binary data.

- 5 The bitstream obtained in the present application may be transmitted quick to obtain the low latency while obtaining high level of compression.

According to an exemplary embodiment, the method further comprises: transmitting the bitstream to a cloud.

- 10 According to a second aspect of the present application, there is provided a method of decoding a point cloud from a bitstream of encoded point cloud data representing a physical object. The method comprises decoding at least one order index difference representing a difference between order indices of two consecutive ordered points based on at least one binary data decoded from the bitstream, each binary data being decoded by: selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already decoded point; context-based
15 entropy decoding said at least one binary data based on the selected context and a probability information relative to the binary data decoded from the bitstream; and decoding an order index difference from the at least one context-based entropy decoded binary data.

- 20 According to an exemplary embodiment, the already decoded point is a penultimate already decoded point with a same sensor index as a sensor index associated with the binary data.

- 25 According to an exemplary embodiment, a context for decoding a binary data being selected from a context table indexed by a context index, and the context index for a binary data equals the distance between the azimuthal angle associated with the binary data and an azimuthal angle of the already decoded point.

According to an exemplary embodiment, a context for decoding a binary data being selected from a context table indexed by a context index and wherein the context index for a binary data depends on a particular rank of said binary data in a series of binary data representing the order index difference.

- 30 According to an exemplary embodiment, a context for decoding a binary data being selected from a context table indexed by a context index, and wherein the context index for a binary data depends on a sensor index associated with the binary data.

- 35 According to an exemplary embodiment, a context for decoding a binary data being selected from a context table indexed by a context index, and the context index for a binary data depends on sensor packs regrouping sensors with close elevation angles.

According to an exemplary embodiment, the method further comprises: obtaining the bitstream from a cloud.

According to an exemplary embodiment, the method further comprises: obtaining the bitstream from a car.

According to a third aspect of the present application, there is provided an apparatus of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object. The apparatus comprising one or more processors configured to carry out a method according to the first aspect of the present application.

- 5 According to a fourth aspect of the present application, there is provided an apparatus of decoding a point cloud from a bitstream of encoded point cloud data representing a physical object. The apparatus comprising one or more processors configured to carry out a method according to the second aspect of the present application.

- 10 According to a fifth aspect of the present application, there is provided a bitstream of encoded point cloud data representing a physical object. The bitstream comprises probability information relative to at least one binary data representing an order index difference representing a difference between order indices of two consecutive ordered points of the point cloud.

- 15 According to a sixth aspect of the present application, there is provided a computer program product including instructions which, when the program is executed by one or more processors, causes the one or more processors to carry out a method according to the first aspect of the present application.

- 20 According to a seventh aspect of the present application, there is provided a non-transitory storage medium carrying instructions of program code for executing a method according to the first aspect of the present application.

According to an eighth aspect of the present application, there is provided a computer program product including instructions which, when the program is executed by one or more processors, causes the one or more processors to carry out a method according to the second aspect of the present application.

- 25 According to a ninth aspect of the present application, there is provided a non-transitory storage medium carrying instructions of program code for executing a method according to the second aspect of the present application.

- 30 The specific nature of at least one of the exemplary embodiments as well as other objects, advantages, features and uses of said at least one of exemplary embodiments will become evident from the following description of examples taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

- 35 Reference will now be made, by way of example, to the accompanying drawings which show exemplary embodiments of the present application, and in which:

Figure 1 illustrates a portion of a predictive tree used for encoding a point cloud in accordance with prior art;

Figure 2 illustrates a linear prediction in a portion of a predictive tree used for encoding a point cloud in accordance with prior art;

Figure 3 illustrates a parallelogram prediction in a portion of a predictive tree used for encoding a point cloud in accordance with prior art;

Figure 4 illustrates a side view of a sensors head and some of its parameters in accordance with prior art;

5 **Figure 5** illustrates a top view of the sensors head and some of its parameters in accordance with prior art;

Figure 6 illustrates a regular distribution of data captured by a spinning sensors head in accordance with prior art;

10 **Figure 7** illustrates a representation of a point in a 3D space in accordance with prior art;

Figure 8 illustrates the encoding of a predictive tree in accordance with prior art;

Figure 9 illustrates a block diagram of steps of a method 100 of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object in accordance with at least one exemplary embodiment;

15 **Figure 10** illustrates an example of encoded ordered points in accordance with one exemplary embodiment of the present application;

Figure 11 illustrates an example of ordered captured point represented in a 2D space in accordance with one exemplary embodiment of the present application;

20 **Figure 12** illustrates another example of ordered captured point represented in a 2D space in accordance with one exemplary embodiment of the present application;

Figure 13 illustrates an example of captured points in accordance with one exemplary embodiment of the present application;

Figure 14 illustrates an example of ordered and quantized captured points in accordance with one exemplary embodiment of the present application;

25 **Figure 15** illustrates a block diagram of steps of a method 200 of decoding a point cloud from a bitstream of encoded point cloud data representing a physical object in accordance with at least one exemplary embodiment;

Figure 16 illustrates a block diagram of steps of a method 800 of entropy-coding order index differences in accordance with at least one exemplary embodiment;

30 **Figure 17** illustrates a block diagram of a context-adaptive arithmetic encoders in accordance with at least one exemplary embodiment;

Figure 18 illustrates drawbacks when a distance between an azimuthal angle associated with the binary data and an azimuthal angle of the last already encoded point;

35 **Figure 19** illustrates drawbacks when a distance between an azimuthal angle associated with the binary data and an azimuthal angle of the last already encoded point;

Figure 20 shows a block diagram of steps of a method 900 of entropy-decoding order index difference in accordance with at least one exemplary embodiment; and

Figure 21 illustrates a schematic block diagram of an example of a system in which various aspects and exemplary embodiments are implemented.

Similar reference numerals may have been used in different figures to denote similar components.

5

DESCRIPTION OF EXEMPLARY EMBODIMENTS

At least one of the exemplary embodiments is described more fully hereinafter with reference to the accompanying figures, in which examples of at least one of the exemplary embodiments are illustrated. An exemplary embodiment may, however, be embodied in many alternate forms and should not be construed as limited to the examples set forth herein. Accordingly, it should be understood that there is no intent to limit exemplary embodiments to the particular forms disclosed. On the contrary, the disclosure is intended to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present application.

When a figure is presented as a flow diagram, it should be understood that it also provides a block diagram of a corresponding apparatus. Similarly, when a figure is presented as a block diagram, it should be understood that it also provides a flow diagram of a corresponding method/process.

At least one of the aspects generally relates to point cloud encoding and decoding, and at least one other aspect generally relates to transmitting a bitstream generated or encoded.

Moreover, the present aspects are not limited to MPEG standards such as MPEG-I part 5 or part 9 that relate to the Point Cloud Compression, and may be applied, for example, to other standards and recommendations, whether pre-existing or future-developed, and extensions of any such standards and recommendations (including MPEG-I part 5 and part 9). Unless indicated otherwise, or technically precluded, the aspects described in the present application may be used individually or in combination.

Figure 9 illustrates a block diagram of steps of a method 100 of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object in accordance with at least one exemplary embodiment.

Geometry data of the point cloud, means the 3D locations of points of the point cloud, is captured by a spinning sensor head.

The spinning sensors head may be the spinning Lidar head 10 comprising multiple lasers (sensors) as explained above. But the scope of the disclosure is not limited to a spinning Lidar head and may apply to any sensors head able to spin around an axis and to capture 3D locations points representing a physical object for each capture angular angle. Sensors may be cameras, depth sensors, lasers, Lidars or scanners.

The captured 3D locations are represented in a 2D coordinates (ϕ, λ) system, as depicted on **Figure 7** together with radius values r_{2D} or r_{3D} . The coordinate ϕ is the azimuthal angle of the sensor head's spin whose discrete values are denoted ϕ_i ($\forall i = 0$ to $I - 1$).

The coordinate λ is a sensor index whose discrete values are denoted λ_k ($\forall k = 0$ to $K - 1$). The radius r_{2D} or r_{3D} belongs to a continuous range of values.

Due to the regular spin (rotation) of the sensors head and the continuous capture with fixed time interval, the azimuthal distance between two points probed by a same sensor is a multiple of an elementary azimuthal shift $\Delta\phi$ as depicted on **Figure 11**. Then, for example, at a first captured time t_1 , five points $P_1(t_1), \dots, P_k(t_1), \dots, P_5(t_1)$ are probed by the five sensors of the Lidar head 10 of **Figure 4** with an azimuthal angle ϕ_1 , at a second capture time t_2 , five points $P_1(t_2), \dots, P_k(t_2), \dots, P_5(t_2)$ are probed by the sensors of the Lidar head 10 with an azimuth angle $\phi_2 = \phi_1 + \Delta\phi$, and so on. Consequently, the discrete value ϕ_1 may be seen as the quantized value of the azimuthal angles ϕ of the points $P_1(t_1), \dots, P_k(t_1), \dots, P_5(t_1)$; quantization being obtained by a quantization step $\Delta\phi$. Similarly, the discrete value ϕ_2 may be seen as the quantized value of the azimuthal angles ϕ of the points $P_1(t_2), \dots, P_k(t_2), \dots, P_5(t_2)$.

In step 110, for each point P_n of the point cloud, a sensor index $\lambda_{k(n)}$ (among the set of sensor indices λ_k ($\forall k = 0$ to $K - 1$)) associated with a sensor that captured the point P_n , an azimuthal angle $\phi_{i(n)}$ (among the set of discrete angles ϕ_i ($\forall i = 0$ to $I - 1$)) representing a capture angle of said sensor and a radius value r_n of spherical coordinates of the point P_n are obtained. For sake of simplicity, $\lambda_{k(n)}$ and the index $i(n)$ will be respectively denoted λ_n and ϕ_n hereafter. Consequently, ϕ_n is not an angle but an index i ($\forall i = 0$ to $I - 1$) pointing to an angle ϕ_i . Nevertheless, because there is an unambiguous relation between the index ϕ_n and the canonically associated azimuthal angle $\phi_{i(n)} = \phi_{\phi_n}$, the quantity ϕ_n is still referred as an azimuthal angle.

According to an exemplary embodiment of step 110, the sensor index λ_n and the azimuthal angle ϕ_n are obtained by converting 3D cartesian coordinates (x_n, y_n, z_n) representing the 3D location of a captured point P_n . These 3D cartesian coordinates (x_n, y_n, z_n) may be output of the sensors head.

In step 120, points of the point cloud are ordered based on the azimuthal angles ϕ_n and the sensor indices λ_n .

According to an exemplary embodiment of step 120, the points are ordered according to a lexicographic order based first on the azimuthal angle and then on the sensor index. Referring back to **Figure 11**, the ordered captured points are $P_1(t_1), \dots, P_k(t_1), \dots, P_5(t_1), P_1(t_2), \dots, P_k(t_2), \dots, P_5(t_2), \dots, P_1(t_n), \dots, P_k(t_n), \dots, P_5(t_n)$. The order index $o(P_n)$ of a point P_n is obtained by:

$$o(P_n) = \phi_n * K + \lambda_n$$

According to another exemplary embodiment of step 120, illustrated on **Figure 12**, the points are ordered according to a lexicographic order based first on the sensor index and then on the azimuthal angle.

The order index $o(P_n)$ of a point P_n is obtained by:

$$o(P_n) = \lambda_n * I + \phi_n$$

In step 130, order index differences Δo_n representing, each, a difference between order

indices of two consecutive points P_{n-1} and P_n (for $n = 2$ to N), are obtained by:

$$\Delta o_n = o(P_n) - o(P_{n-1})$$

Encoding ordered points into a bitstream B may comprise encoding at least one order index difference Δo_n . Optionally, it may also comprise encoding radius values r_n (essentially representative of either r_{2D} or r_{3D} of the point P_n), cartesian residuals ($x_{res,n}, y_{res,n}, z_{res,n}$) of three-dimensional cartesian coordinates of ordered points and angular residual $\phi_{res,n}$.

The order index $o(P_1)$ of the first point P_1 may be directly coded into the bitstream B. This is equivalent to arbitrary setting the order index of a virtual zero-th point to zero, i.e. $o(P_0) = 0$, and coding $\Delta o_1 = o(P_1) - o(P_0) = o(P_1)$.

Given the order index $o(P_1)$ of the first point and the order differences Δo_n , one can recursively reconstruct the order index $o(P_n)$ of any point P_n by:

$$o(P_n) = o(P_{n-1}) + \Delta o_n$$

Then, sensor indices λ_n and azimuthal angle ϕ_n are obtained by:

$$\lambda_n = o(P_n) \text{ modulo } K \quad (1)$$

$$\phi_n = o(P_n) / K \quad (2)$$

where the division $/ K$ is the integer division (aka Euclidian division). Therefore, $o(P_1)$ and Δo_n are an alternative representation of λ_n and ϕ_n .

In step 140, the order index $o(P_n)$ associated with ordered points are encoded, in the bitstream B, by encoding (N-1) order index differences Δo_n ($\forall n = 2 \text{ to } N$) where N is the number of ordered points. Each order index $o(P_n)$ represents a difference between order indices associated with two consecutive ordered points. On **Figure 10**, five ordered points are represented (black circles): two points P_n and P_{n+1} were captured in time t1 with an angular angle ϕ_c (among the ϕ_i 's) and three points were captured in time t2 with an angular angle $\phi_c + \Delta\phi$. Assuming the coordinates of the first point P_n in the 2D coordinates (ϕ, λ) system are known beforehand, a first order index difference Δo_{n+1} is obtained as a difference between the order index $o(P_{n+1})$ associated with the point P_{n+1} and the order index $o(P_n)$ associated with the point P_n . A second order index difference Δo_{n+2} is obtained as a difference between the order index $o(P_{n+2})$ associated with another ordered point P_{n+2} and the order index $o(P_{n+1})$ associated with P_{n+1} , and so on.

Ordering the captured points provides interaction between points captured by different sensors of the spinning sensors head. A single encoding is thus required for encoding those ordered points leading to a very simple and low latency encoding.

Reconstructing points from the order index differences Δo_n requires information such as a number N of points of the point cloud, an order index $o(P_1)$ of a first point in the 2D coordinates (ϕ, λ) system, and sensor setup parameters such as the elementary azimuthal shift $\Delta\phi$ or an elevation angle θ_n associated with each sensor. This information may also be encoded in the bitstream B or signaled by another mean or may be known beforehand by a decoder.

The order index differences Δo_n are entropy-encoded as explained later in reference with **Figure 16**.

Optionally, the method further comprises in step 150, encoding, in the bitstream B, the radius values r_n of spherical coordinates associated with ordered points of the point cloud.

5 According to an exemplary embodiment of step 150, the radius values r_n are quantized.
According to an exemplary embodiment of step 150, the radius values r_n are quantized.
According to an exemplary embodiment of step 150, the radius values r_n are quantized and entropy coded.

10 According to an exemplary embodiment of step 150, the radius values r_n represents the radius r_{3D} .

According to an exemplary embodiment of step 150, the radius values r_n represents the projected radius r_{2D} on the horizontal xy plane as depicted on **Figure 7**.

15 Optionally, the method further comprises, in step 160, encoding the residuals $(x_{res,n}, y_{res,n}, z_{res,n})$ of three-dimensional cartesian coordinates of ordered points P_n based on their three-dimensional cartesian coordinates (x_n, y_n, z_n) , on decoded azimuthal angles $\phi_{dec,n}$, decoded radius values $r_{dec,n}$ obtained from radius values r_n and on sensor indices λ_n .

20 According to an exemplary embodiment of step 160, the residuals $(x_{res,n}, y_{res,n}, z_{res,n})$ are differences between the three-dimensional cartesian coordinates (x_n, y_n, z_n) of the points of the point cloud and estimated three-dimensional coordinates $(x_{estim,n}, y_{estim,n}, z_{estim,n})$.

According to an exemplary embodiment of step 160, the residuals $(x_{res,n}, y_{res,n}, z_{res,n})$ are given by:

$$\begin{cases} x_{res,n} = x_n - x_{estim,n} \\ y_{res,n} = y_n - y_{estim,n} \\ z_{res,n} = z_n - z_{estim,n} \end{cases}$$

25 According to an exemplary embodiment of step 160, the estimated coordinates $(x_{estim,n}, y_{estim,n})$ associated with an ordered point P_n are based on decoded azimuthal angles $\phi_{dec,n}$ and decoded radius values $r_{dec,n}$ associated with the point P_n .

According to an exemplary embodiment of step 160, the residuals $(x_{res,n}, y_{res,n}, z_{res,n})$ are entropy coded.

30 According to an exemplary embodiment of step 160, the estimated coordinates $(x_{estim,n}, y_{estim,n})$ are given by:

$$\begin{cases} x_{estim,n} = r_{dec,n} * \cos(\phi_{dec,n}) \\ y_{estim,n} = r_{dec,n} * \sin(\phi_{dec,n}) \end{cases}$$

According to an exemplary embodiment of a step 160, the estimated coordinate $(z_{estim,n})$ associated with an ordered point is based on a decoded radius value $r_{dec,n}$ associated with the point and an elevation angle θ_n of a sensor that captured the point.

35 According to an exemplary embodiment of step 160, the estimated coordinates $(z_{estim,n})$ are also based on the sensor indices λ_n .

According to an exemplary embodiment of step 160, the estimated coordinates $(z_{estim,n})$

are given by:

$$z_{estim,n} = r_{dec,n} \tan \theta_n$$

Optionally, the method further comprises, in step 170, encoding, in the bitstream B, residual azimuthal angles $\phi_{res,n}$ associated with ordered points. According to an exemplary embodiment of a step 170, the azimuthal angles ϕ_n are quantized by:

$$\phi_n = \text{round}(\phi(P_n)/\Delta\phi)$$

where $\phi(P_n)$ is the original azimuthal angle of point P_n . In this case, the set of discrete angles ϕ_i ($0 \leq i < I$) is essentially defined by $\phi_i = i * \Delta\phi$ and one obtains $\phi_{i(n)} = \phi_n * \Delta\phi$.

The order index $o(P_n)$ of a point P_n is thus given by:

$$o(P_n) = \phi_n * K + \lambda_n = \text{round}(\phi(P_n)/\Delta\phi) * K + \lambda_n.$$

The residual azimuthal angles $\phi_{res,n}$ is given by:

$$\phi_{res,n} = \phi(P_n) - \phi_n * \Delta\phi \quad (3)$$

This exemplary embodiment of step 170 provides advantages because sometimes, in practice, not all points are captured at each capture time because a noise may be captured or because sensors may not be all perfectly aligned or because at least one laser beam of a Lidar sensors head may not be reflected. Captured points may then look as depicted on **Figure 13**. Quantizing the azimuthal angles $\phi(P_n)$ leads to an easier discrete representation of points in the 2D coordinates (ϕ, λ) system as depicted on **Figure 14** that allows an easier path for ordering the points of the point cloud.

The residual azimuthal angles $\phi_{res,n}$ are encoded into the bitstream B, preferably by quantizing and/or entropy-coding.

This exemplary embodiment of step 170 also reduces the dynamic of the angular angles to be encoded into the bitstream because only a residual is encoded rather than the full range value. High compression performance is reached.

Optionally, the method further comprises, in step 180, the decoded azimuthal angles $\phi_{dec,n}$ are obtained based on the azimuthal angle ϕ_n .

According to an embodiment of step 180, the decoded azimuthal angles $\phi_{dec,n}$ is given by:

$$\phi_{dec,n} = \phi_n * \Delta\phi$$

According to an embodiment of step 180, the decoded azimuthal angles $\phi_{dec,n}$ are obtained based on azimuthal angle ϕ_n , the elementary azimuthal shift $\Delta\phi$ and residual azimuthal angles $\phi_{res,n}$.

According to an embodiment of step 180, the decoded azimuthal angles $\phi_{dec,n}$ is given by:

$$\phi_{dec,n} = \phi_n * \Delta\phi + \phi_{res,n}$$

According to an embodiment of step 180, the decoded azimuthal angles $\phi_{dec,n}$ are obtained based on the azimuthal angle ϕ_n , the elementary azimuthal shift $\Delta\phi$ and a decoded angular residual $\phi_{dec,res,n}$ obtained by de-quantizing the quantized residual azimuthal angle $\phi_{res,n}$ given by equation 3.

According to an embodiment of step 180, the decoded azimuthal angles $\phi_{dec,n}$ is given by:

$$\phi_{dec,n} = \phi_n * \Delta\phi + \phi_{dec,res,n}$$

5 Optionally, in step 190, the decoded radius values $r_{dec,n}$ are obtained based on encoded radius values r_n .

According to an exemplary embodiment of step 190, the decoded radius values $r_{dec,n}$ is obtained by de-quantizing a quantized radius values r_n .

10 **Figure 15** illustrates a block diagram of steps of a method 200 of decoding a point cloud from a bitstream of encoded point cloud data representing a physical object in accordance with at least one exemplary embodiment.

Decoding points of a point cloud from a bitstream B requires information such a number N of points of the point cloud, order index $o(P_1)$ of a first point in the 2D coordinates (ϕ, λ) system, and sensor setup parameters such as the elementary azimuthal shift $\Delta\phi$ or an elevation angle θ_k associated with each sensor k . This information may also be decoded from the bitstream B or received by any other means or may be known beforehand by a decoder.

15 N points of the point cloud are decoded recursively.

In step 210, at least one order index difference Δo_n ($n = 2$ to N) is decoded from the bitstream B. Each order index difference Δo_n is decoded for a current point P_n .

20 In step 220, an order index $o(P_n)$ is obtained for a current point P_n by:

$$o(P_n) = o(P_{n-1}) + \Delta o_n$$

The order index difference Δo_n represents a difference between an order index associated with the current point P_n and another order index $o(P_{n-1})$ associated with a (already) previously decoded point P_{n-1} .

25 In step 230, a sensor index λ_n associated with a sensor that captured the current point P_n and an azimuthal angle ϕ_n representing a capture angle of said sensor are obtained from the order index $o(P_n)$.

According to an exemplary embodiment of step 230, the sensor index λ_n and the azimuthal angles ϕ_n are obtained by equations (1) and (2).

30 Optionally, in step 240, a decoded azimuthal angle $\phi_{dec,n}$ is obtained based on the azimuthal angle ϕ_n .

According to an embodiment of step 240, the decoded azimuthal angles $\phi_{dec,n}$ is obtained based on the azimuthal angle ϕ_n and the elementary azimuthal shift $\Delta\phi$.

According to an embodiment of step 240, the decoded azimuthal angle $\phi_{dec,n}$ is given by:

35
$$\phi_{dec,n} = \phi_n * \Delta\phi$$

According to an embodiment of step 240, the decoded azimuthal angle $\phi_{dec,n}$ is obtained based on a residual azimuthal angle $\phi_{res,n}$ decoded from the bitstream B.

According to an embodiment of step 240, the decoded azimuthal angle $\phi_{dec,n}$ is given by:

$$\phi_{dec,n} = \phi_n * \Delta\phi + \phi_{res,n}$$

Optionally, in step 250, a radius value r_n of spherical coordinates of the current points P_n is decoded from the bitstream B.

5 According to an exemplary embodiment of step 250, the radius values r_n are de-quantized to obtain decoded radius values $r_{dec,n}$.

According to an exemplary embodiment of step 250, the radius values r_n are entropy-decoded and de-quantized to obtain decoded radius values $r_{dec,n}$.

Optionally, in step 260, a residual $(x_{res,n}, y_{res,n}, z_{res,n})$ of three-dimensional cartesian coordinates of the current point P_n is decoded from a bitstream B.

10 According to an exemplary embodiment of step 260, the residual $(x_{res,n}, y_{res,n}, z_{res,n})$ is entropy-decoded.

Optionally, in step 270, the three-dimensional cartesian coordinates (x, y, z) of the current point P_n are decoded based on the decoded residual $(x_{res,n}, y_{res,n}, z_{res,n})$ of the three-dimensional cartesian coordinates of the current point P_n , the radius value r_n , the decoded azimuthal angle $\phi_{dec,n}$ and the sensor index λ_n .

According to an exemplary embodiment of step 270, the three-dimensional cartesian coordinates (x, y, z) of the current point P_n are the sums of the residuals $(x_{res,n}, y_{res,n}, z_{res,n})$ and estimated three-dimensional coordinates $(x_{estim}, y_{estim}, z_{estim})$:

$$\begin{cases} x = x_{res,n} + x_{estim} \\ y = y_{res,n} + y_{estim} \\ z = z_{res,n} + z_{estim} \end{cases}$$

20 According to an exemplary embodiment of step 270, the estimated coordinates (x_{estim}, y_{estim}) associated with the current point P_n are based on the azimuthal angle ϕ_n and the radius value r_n .

According to an exemplary embodiment of step 270, the estimated coordinates (x_{estim}, y_{estim}) are given by

$$\begin{cases} x_{estim} = r_n * \cos(\phi_{\phi_n}) \\ y_{estim} = r_n * \sin(\phi_{\phi_n}) \end{cases}$$

According to another exemplary embodiment of step 270, the estimated coordinates $(x_{estim,n}, y_{estim,n})$ are given by:

$$\begin{cases} x_{estim,n} = r_{dec,n} * \cos(\phi_{dec,n}) \\ y_{estim,n} = r_{dec,n} * \sin(\phi_{dec,n}) \end{cases}$$

30 where $r_{dec,n}$ are decoded radius values obtained from radius values r_n . For example, the decoded radius values $r_{dec,n}$ may be obtained by de-quantizing the radius values r_n .

According to an exemplary embodiment of step 270, the estimated coordinate (z_{estim}) associated with the current point P_n is based on the radius value r_n associated with current point P_n and an elevation angle θ_k of a sensor k that captured the current point P_n .

35 According to an exemplary embodiment of step 270, the estimated coordinates (z_{estim}) are given by:

$$z_{estim} = r_n \tan \theta_k$$

According to an exemplary embodiment of step 270, the estimated coordinates (z_{estim}) are also based on the sensor index λ_n .

- 5 According to an exemplary embodiment of step 270, the estimated coordinates (z_{estim}) are given by:

$$z_{estim} = r_n \tan \theta_{\lambda_n}$$

Figure 16 shows a block diagram of steps of a method 800 of entropy-coding an order index difference Δo_n in accordance with at least one exemplary embodiment.

- 10 In step 810, at least one binary data f_j is obtained by binarizing the at least one order index difference Δo_n .

- For each binary data f_j , a context is selected (step 820) based on a distance C_j between an azimuthal angle ϕ^j associated with the binary data f_j and an azimuthal angle of an already encoded point, and each binary data f_j is context-adaptive entropy coded (830) in the bitstream B, based on the selected context. As will become clear hereafter, similarly to the quantity ϕ_n , the quantity ϕ^j is abusively referred as an azimuthal angle even if it is an index between 0 and I-1 that points to a particular discrete angle among the ϕ_i 's.

- 15 Context-adaptive entropy encoding the order index difference provides efficient compression performance compared to other encoding technics such as a predictive tree encoding technique because such prediction-based encoding technics do not catch the intrinsic structure of the representation (ϕ, λ) of point locations. Such structure is captured by the exemplary embodiments of the present application by selecting contexts and entropy encoding the order index difference based on the selected context.

- 20 According to an exemplary embodiment of step 810, the binarization of an order index difference Δo_n is performed by a series of binary data f_j , for $j \geq 0$. A binary data f_j equals to a particular value PV (for example $f_j=1$) to indicate if Δo_n is equal to j and does not equal to said specific value PV otherwise (for example $f_j=0$). Then, a first binary f_0 is coded. For example, if $f_0=PV$, the coding of Δo_n is finished as $\Delta o_n=0$ has been coded; otherwise, if $f_0 \neq PV$, then a second binary data f_1 is coded. If $f_1 \neq PV$, then a third binary data f_2 is coded and so on. This is basically a unary coding of Δo_n .

- 30 According to an exemplary embodiment of step 810, when the points are ordered according to a lexicographic order based first on the azimuthal angle and then on the sensor index (**Figure 11**), the azimuthal angle ϕ^j associated with a binary data f_j is the azimuthal angle ϕ_{n-1} associated with a current point P_{n-1} if $j + c < K$ where c is the index of the current point modulo K (or equivalently c is the sensor index λ_{n-1} of the current point); otherwise the azimuthal angle ϕ^j associated with a binary data f_j is the azimuthal angle

$$\phi^j = \phi_n + s$$

- 40 where s is an integer such that $sK \leq j + c < (s + 1)K$. Each binary data f_j is also associated with a sensor index λ^j which corresponds to the sensor index of a sensor that would have captured the next ordered point P_n in case f_j would be equal to PV, i.e. $\lambda^j = \lambda_{n-1} + j \bmod K$.

On **Figure 11**, the order index difference Δo_n equals 8 and is binarized into nine binary data f_0 to f_8 all not equal to PV except the last one f_8 which equals to PV. Azimuthal angles ϕ^0 to ϕ^6 respectively associated with binary data f_0 to f_6 are equal to the azimuthal angle ϕ_{n-1} and azimuthal angles ϕ^7 and ϕ^8 respectively associated with f_7 and f_8 are equal to ϕ_n .

According to an exemplary embodiment of step 810, when the points are ordered according to a lexicographic order based first on the sensor index and then on the azimuthal angle (**Figure 12**), the azimuthal angles ϕ^j associated with a binary data f_j is given by:

$$\phi^j = \phi_{n-1} + j$$

Each binary data f_j is also associated with a sensor index λ^j which corresponds to the sensor index of a sensor that would have captured the next ordered point P_n in case f_j would be equal to PV, i.e. $\lambda^j = \lambda_{n-1}$.

On **Figure 12**, the order index difference Δo_n equals 4 and is binarized into five binary data f_0 to f_4 all not equal to PV except the last one f_4 which equals to PV. Azimuthal angles ϕ^0 to ϕ^4 respectively associated with binary data f_0 to f_4 are respectively ϕ_{n-1} , $\phi_{n-1} + 1$, $\phi_{n-1} + 2$, $\phi_{n-1} + 3$ and $\phi_{n-1} + 4 = \phi_n$.

Optionally, in step 840, a residual R is obtained as a difference between the order index difference Δo_n and a maximum number of binary data (upper bound) N_{flag} . Then, if a binary data $f_{N_{\text{flag}}-1}$ is reached and is not equal to PV, then one has necessarily $\Delta o_n \geq N_{\text{flag}}$ and the residual R is given by: $R = \Delta o_n - N_{\text{flag}}$.

The maximum number of flags N_{flag} is set to bound the number of arithmetically coded binary information per point and then bound the latency to obtain next point.

According to an embodiment of step 840, the residual R is encoded by using an exp-Golomb code.

According to an exemplary embodiment of step 810, a binary data f_j is a flag and $PV = 1$.

According to an exemplary embodiment of step 830, a binary data f_j is entropy coded by a Context-Adaptive Binary Arithmetic Coder (like CABAC).

Figure 17 illustrates schematically a block diagram of a Context-Adaptive Binary Arithmetic encoder.

Firstly, a context is selected by some selection process based on already coded information to provide a context index ctxIdx associated with each binary data f_j (binary symbol). A context table with N_{ctx} entries stores the probabilities associated with the contexts and the probability p_{ctxIdx} is obtained as the ctxIdx -th entry of the context table. A binary symbol f_j is encoded in the bitstream by an entropy coder using the probability p_{ctxIdx} .

Entropy coders are usually arithmetic coders but may be any other type of entropy coders like asymmetric numeral systems. In any case, optimal coders adds $-\log_2(p_{\text{ctxIdx}})$ bits in the bitstream to encode $f_j = 1$ or $-\log_2(1 - p_{\text{ctxIdx}})$ bits in the bitstream to encode $f_j = 0$. Once the symbol f_j is encoded, the probability p_{ctxIdx} is updated by using an update process

taking f_j and p_{ctxidx} as entries; the update process is usually performed by using update tables. The updated probability replaces the $ctxidx$ -th entry of the context table. Then, another symbol can be encoded, and so on. The update loop back to the context table is a bottleneck in the coding workflow as another symbol can be encoded only after the update has been performed. For this reason, the memory access to the context table must be as quick as possible and minimizing the size of the context table helps easing its hardware implementation.

A Context-Adaptive Binary Arithmetic decoder performs essentially the same operations the Context-Adaptive Binary Arithmetic encoder except that the coded symbol f_j is decoded from the bitstream by an entropy decoder using the probability p_{ctxidx} .

The choice of the adequate context, i.e. the probability p_{ctxidx} that estimates at best the chance of the binary data f_j to be equals to PV, is essential to obtain good compression. Therefore, the context selection should use relevant already coded information and correlation to obtain this adequate context.

According to an exemplary embodiment of step 820, the distance C_j associated with a binary data f_j depends on an azimuthal angle ϕ^j associated with the binary data f_j and an (index of an) azimuthal angle ϕ_{penult} associated with a penultimate already coded point P_{penult} with a same sensor index as the sensor index λ^j associated with the binary data f_j . The current point P_{n-1} is never considered in the set of already coded points from which the penultimate already coded point P_{penult} is determined.

According to an exemplary embodiment of step 820, the distance C_j associated with a binary data f_j is given by:

$$C_j = \phi^j - \phi_{penult,j} \quad (13)$$

It is reminded that the two quantities ϕ^j and $\phi_{penult,j}$ are indices pointing to discrete angles in the set of discrete number of values ϕ_i ($i = 0$ to $I - 1$).

On **Figure 11**, the distance $C_0 = 0$ because last and penultimate already coded point with sensor index equals to $\lambda^0 = \lambda_{n-1} = 2$, are located at the same angular location $\phi^0 = \phi_{n-1}$ as the current point P_{n-1} ; this may happen in some specific sensor configuration. Then, $C_1 = 4$ because the difference between, on one hand, the azimuthal angles $\phi_{penult,3}$, associated with the penultimate already coded point (grey circle) with the sensor index $\lambda^1 = \lambda_{n-1} + 1 = 3$ associated with f_1 , and, on the other hand, $\phi^1 = \phi_{n-1}$ equals 4. Then, $C_2 = 3$ because the difference between the azimuthal angles $\phi_{penult,4}$ associated with the penultimate already coded point with the sensor index $\lambda^2 = \lambda_{n-1} + 2 = 4$ associated with f_2 , and $\phi^2 = \phi_{n-1}$ equals 3, and so on. Further, $C_7 = 3$ because the difference between the azimuthal angles $\phi_{penult,0}$ associated with the penultimate already coded point with the sensor index $\lambda^7 = \lambda_{n-1} + 7 = 9 = 0 \bmod K$ (here $K=9$) associated with f_7 and $\phi^7 = \phi_{n-1} + 1$ equals 3. Finally, $C_8 = 2$ because the difference between the azimuthal angles $\phi_{penult,1}$ associated with the penultimate already coded point with the sensor index $\lambda^8 = \lambda_{n-1} + 8 = 10 = 1 \bmod K$ associated with f_8 and $\phi^8 = \phi_{n-1}$ equals 2.

On **Figure 12**, all binary data f_j are associated with the same sensor index $\lambda^j = \lambda_{n-1}$. The distance $C_0 = 2$ because the difference between the azimuthal angle

$\phi_{penult, \lambda_{n-1}}$ associated with the penultimate already coded point (grey circle) and $\phi^0 = \phi_{n-1}$ equals 2. The distance $C_1=3$ because the difference between the azimuthal angles $\phi_{penult, \lambda_{n-1}}$ associated with the penultimate already coded point and $\phi^1 = \phi_{n-1} + 1$ equals 3, and so on.

- 5 The dependency of the distance C_j with the penultimate already coded point P_{penult} provides advantages compared to other alternative dependencies such as a dependency with the last, the three-last or any other last already encoded point for the following reasons given in relation with **Figures 18 and 19**.

10 When probing a physical object, a probing laser typically provides a reflected signal for several successive captured angles. In this case, when the points are ordered according to a lexicographic order based first on the sensor index and then on the azimuthal angle (**Figure 12**), one expects to obtain several successive order index difference Δo_n equal to one as shown on **Figure 18**.

15 The upper row depicts points P_n located at their angle position $\phi = \phi(P_n)$ and the lower row shows the same points after quantization of ϕ into $\phi_c = \phi_{i(n)} = \phi_n * \Delta\phi$. Each quantization interval is occupied by one and only point, leading to successive order index difference $\Delta o_n = +1$. However, when sensing noise is added, some points may jump from one interval to an adjacent one, as seen on **Figure 19** where the second point has moved slightly to the left. Consequently, the Δo_n pattern has changed from +1 +1 +1 +1 to +1 0 +2 +1. Similarly, it may also change to +1 +2 +0 +1 if a point moves to the right.

20 The distance C_j (equation 13) can detect this pattern and anticipate the following Δo_n . Basically, if C_j is equal to 2 (obtained from patterns +1 +1, +2 0 or +0 +2), it is likely that the point corresponding to $f_j = PV$ is present and the context corresponding to $C_j=2$ will provide a high probability of coding PV as value for f_j . On the other hand, if C_j is equal to 25 1 (obtained from +1 +0), one expects to compensate for "being late" and it is more likely that the point corresponding to $f_j = PV$ is not present and the context corresponding to $C_j=1$ will provide a low probability of coding PV as value for f_j .

30 Clearly, and shown by the tests, a distance C_j given by $\phi^j - \phi_{last, j}$, means using the last point instead of the penultimate point is less efficient in anticipating Δo_n . For example, either a +1 or a 0 can follow a +2 in the patterns +1 0 +2 +1 and +1 +2 0 +1.

35 Tests have been performed on one frame of a QNX test sequence and of a Ford test sequence taken from the MPEG test set for G-PCC. The azimuthal angles and the sensor indices representing the points captured by a spinning sensors head have been encoded by 1) the G-PCC point cloud encoding with a 2D input point cloud formed by the azimuthal angles and the sensor indices, 2) encoding the azimuthal angles and the sensor indices by entropy-encoding order index differences Δo_n with contexts that do not depends on the distance C_j , and 3) encoding the azimuthal angles and the sensor indices by entropy-encoding order index differences Δo_n with contexts depends on the distance C_j . In both 2) and 3), the binarization of Δo_n has been performed as described above by using f_j and R.

40 Table 1 shows obtained compression rates when the points are ordered according to a lexicographic order based first on the azimuthal angle and then on the sensor index

(Figure 11), and Table 2 shows obtained compression rates when the points are ordered according to a lexicographic order based first on the sensor index and then on the azimuthal angle (Figure 12). Results are presented as bits per point (bpp) for lossless compression for both lexicographic orders. Lower bpp values indicate better compression.

5

Table 1

Sequence	1)	2)	3)
QNX	0.73813	0.64338	0.58621
Ford	1.5916	1.7655	1.2756

Table 2

Sequence	1)	2)	3)
QNX	0.73813	0.61897	0.58181
Ford	1.5916	1.5374	1.0623

The solution 3) provides 5% to 30% gains of compression and clearly outperforms solution 1 (G-PCC).

10 Implementing the dependency of the distance C_j requires accessing azimuthal angles ϕ_{penult} associated with penultimate already coded point P_{penult} with a same sensor index as the sensor index λ^j of a binary data f_j . Implementing such dependency by using a rolling buffer leads to a low complexity encoding/decoding.

For example, a rolling buffer may be a 2D array storing in a first column the K values $\phi_{penult}(k)$ ($\forall k = 0$ to $K - 1$) of the penultimate coded points with sensor index equal to k , and in another column the K values $\phi_{last}(k)$ ($\forall k = 0$ to $K - 1$) of the last coded points with sensor index equal to k . When, the distance C_j has to be evaluated for a binary data f_j , the azimuthal angle $\phi_{penult}(\lambda^j)$ for a sensor index λ^j (associated with f_j) is directly obtained from the rolling buffer. After each (de)coding of an order index difference Δo_n , the azimuthal angle ϕ_n is obtained and the buffer is rolled as follows

20

$$\phi_{penult}(k) = \phi_{last}(k) \quad \text{and} \quad \phi_{last}(k) = \phi_n$$

where ϕ_n is the azimuthal angle of the next point P_n whose order index is $o(P_n) = o(P_{n-1}) + \Delta o_n$.

25 According to an exemplary embodiment of step 820, the distance C_j associated with a binary data f_j depends on an azimuthal angle ϕ^j associated with the binary data f_j and an (index of an) azimuthal angle ϕ_{al} associated with any of the already coded points not necessarily probed by the same sensor index as the sensor index of a current point P_{n-1} .

According to an exemplary embodiment of step 820, the distance C_i associated with a binary data f_j is given by:

$$C_j = \phi^j - \phi_{al}$$

30

According to an exemplary embodiment of step 820, a context index $ctxIdx$ of a context table associated with a binary data f_j equals to the distance C_j .

According to an exemplary embodiment of step 320, a context index $ctxIdx$ is capped by a threshold $th1$ to limit the number of contexts:

$$ctxIdx = \min(C_j, th1)$$

For example, $th1$ may be equal to 6.

- 5 Limiting the number of contexts is important in order to ease implementation, limit memory use and also to ensure statistical relevance of the probabilities p_{ctxIdx} as high values of j may not be often visited.

- 10 According to an exemplary embodiment of step 820, a context index $ctxIdx$ for a binary data f_j depends on the rank j of said binary data in a series of binary data representing the order index difference Δo_n .

Because the statistics of the binary data f_j tend to become weakly dependent on the rank j for high j , one may take advantage of this weak dependence to limit the number of contexts.

- 15 Moreover, dependency between the context index $ctxIdx$ and the value j provides advantages for the following reasons: the statistics of each binary flag f_j being equal to PV or not depend on the index j . Typically the probability $P(f_j=PV \mid f_0 \text{ to } f_{j-1} \neq PV)$ decreases with j . Therefore, one obtains better probabilities p_{ctxIdx} by making the index $ctxIdx$ depends on j

According to an exemplary embodiment of step 820, a context index $ctxIdx$ is given by:

- 20
$$ctxIdx = \min(C_j, th1) * (th2+1) + \min(j, th2)$$

where th_2 is a second threshold such as to have $(th1+1)*(th2+1)$ contexts. For example, one may take $th2$ equal to 3.

According to an exemplary embodiment of step 820, the context index $ctxIdx$ depends on the sensor index λ associated with the binary data f_j .

- 25 The statistics of each binary data f_j being equal or not to PV also depends on the sensor (laser) index probing the location of where the next point would be, should the binary data f_j equals to PV. For example, sensors directed to the ground almost always probe something (mostly the road), but sensors directed to the sky tend to probe more object less often, like buildings or trees. But dependency between the context index $ctxIdx$ and the sensor index λ^j associated with the binary data f_j multiplies the number of contexts by K (the number of sensors) and likely provides a very high number of contexts as typical Lidar head have 16, 32 or even 64 laser beams.
- 30

- 35 According to an exemplary embodiment of step 820, the sensors with close elevation angles are regrouped together into packs, because all sensors of the same pack tend to probe similar objects.

Each sensor whose index is λ has a corresponding pack index $\lambda_{pack}(\lambda)$, and the binary data f_j associated the sensor index λ^j has an associated pack index with pack index $\lambda_{j,pack}$.

Grouping the sensors into packs limits the number of contexts.

According to an exemplary embodiment of step 820, the context index $ctxIdx$ is given by:

$$ctxIdx = (\min(C_j, th) * (th_2 + 1) + \min(j, th_2)) * N_{pack} + \lambda_{j,pack}$$

where N_{pack} is a number of packs.

The number of contexts becomes $(th+1) * (th_2+1) * N_{pack}$.

- 5 In a variant, a pack index $\lambda_{j,pack}$ is given by:

$$\lambda_{j,pack} = \text{floor}(\lambda^j * N_{pack} / K)$$

such that $\lambda_{j,pack}$ belongs to $[0, N_{pack} - 1]$.

- 10 The number of contexts may then be tuned by setting adequate values to $th1$ and/or th_2 and/or N_{pack} . These values may be coded in the bitstream B. For example, with respective values 6, 3 and 4, one gets $7 * 4 * 4 = 112$ which is an acceptable number of contexts.

Figure 20 shows a block diagram of steps of a method 900 of entropy-decoding an order index difference Δo_n in accordance with at least one exemplary embodiment.

- 15 Decoding, from the bitstream B, a point P_n of a point cloud representing a physical object, comprises decoding, from the bitstream B, an order index difference Δo_n representing a difference between an order index $o(P_n)$ associated with the point P_n and another order index $o(P_{n-1})$ associated with a current point P_{n-1} .

- 20 Decoding an order index difference Δo_n comprises decoding at least one binary data f_j from the bitstream B. For each binary data f_j , the method selects (step 910) a context based on a distance C_j between a azimuthal angle ϕ^j associated with the binary data f_j and an azimuthal angle (ϕ_{penult} or ϕ_{al}) of an already decoded point. The method then context-based entropy decodes (step 920) said at least one binary data f_j based on the selected contexts and probability information relative to the binary data f_j and decoded from the bitstream B. Next, the order index difference Δo_n is obtained (step 930) from the context-based entropy decoded binary data f_j .

- 25 Optionally, in step 930, if a binary data $f_{Nflag-1}$ is reached and is not equal to PV, then a residual R is also decoded from the bitstream B and the order index difference Δo_n is obtained by $\Delta o_n = R + N_{flag}$.

According to an embodiment of step 930, the residual R is decoded by using an exp-Golomb code.

- 30 Context selection of steps 820 and 910 are the same. Therefore, all exemplary embodiments and variants of step 820 apply to step 910.

- 35 According to an exemplary embodiment of step 920, a binary data f_j is entropy decoded by a Context-Adaptive Binary Arithmetic decoder (like CABAC). Context-based entropy decoding (step 920) a binary data f_j based on a selected context is essentially the same as the Context-based entropy encoding on **Figure 17**.

Basically, entropy decoders decode $-\log_2(p_{ctxIdx})$ bits from the bitstream B to decode a binary value $f_j = 1$ or $-\log_2(1 - p_{ctxIdx})$ bits from the bitstream B to decode $f_j = 0$. Once the symbol f_j is decoded, the probability p_{ctxIdx} is updated by using an update process taking

f_j and p_{ctxldx} as entries; the update process is usually performed by using update tables. The updated probability replaces the $ctxldx$ -th entry of the context table. Then, another symbol can be decoded, and so on. The update loop back to the context table is a bottleneck in the coding workflow as another symbol can be decoded only after the update has been performed. For this reason, the memory access to the context table must be as quick as possible and minimizing the size of the context table helps easing its hardware implementation.

According to an embodiment of step 930, the order index difference Δo_n is decoded by a unary decoding of the at least one decoded binary data f_j .

For example, if a first $f_0 = PV$, the decoding is finished as $\Delta o_n = 0$ has been decoded; otherwise, if $f_0 \neq PV$, then a second binary data f_1 is decoded otherwise $\Delta o_n = 1$ and the decoding is finished. If $f_1 \neq PV$, then a third binary data f_2 is decoded and so on.

Figure 21 shows a schematic block diagram illustrating an example of a system in which various aspects and exemplary embodiments are implemented.

System 300 may be embedded as one or more devices including the various components described below. In various embodiments, the system 300 may be configured to implement one or more of the aspects described in the present application.

Examples of equipment that may form all or part of the system 300 include personal computers, laptops, smartphones, tablet computers, digital multimedia set top boxes, digital television receivers, personal video recording systems, connected home appliances, connected vehicles and their associated processing systems, head mounted display devices (HMD, see-through glasses), projectors (beamers), "caves" (system including multiple displays), servers, video encoders, video decoders, post-processors processing output from a video decoder, pre-processors providing input to a video encoder, web servers, set-top boxes, and any other device for processing a point cloud, a video or an image or other communication devices. Elements of system 300, singly or in combination, may be embodied in a single integrated circuit (IC), multiple ICs, and/or discrete components. For example, in at least one embodiment, the processing and encoder/decoder elements of system 300 may be distributed across multiple ICs and/or discrete components. In various embodiments, the system 300 may be communicatively coupled to other similar systems, or to other electronic devices, via, for example, a communications bus or through dedicated input and/or output ports.

The system 300 may include at least one processor 310 configured to execute instructions loaded therein for implementing, for example, the various aspects described in the present application. Processor 310 may include embedded memory, input output interface, and various other circuitries as known in the art. The system 300 may include at least one memory 320 (for example a volatile memory device and/or a non-volatile memory device). System 300 may include a storage device 340, which may include non-volatile memory and/or volatile memory, including, but not limited to, Electrically Erasable Programmable Read-Only Memory (EEPROM), Read-Only Memory (ROM), Programmable Read-Only Memory (PROM), Random Access Memory (RAM), Dynamic Random-Access Memory (DRAM), Static Random Access Memory (SRAM), flash, magnetic disk drive, and/or

optical disk drive. The storage device 340 may include an internal storage device, an attached storage device, and/or a network accessible storage device, as non-limiting examples.

5 The system 300 may include an encoder/decoder module 330 configured, for example, to process data to provide encoded/decoded point cloud geometry data, and the encoder/decoder module 330 may include its own processor and memory. The encoder/decoder module 330 may represent module(s) that may be included in a device to perform the encoding and/or decoding functions. As is known, a device may include one or both of the encoding and decoding modules. Additionally, encoder/decoder module
10 330 may be implemented as a separate element of system 300 or may be incorporated within processor 310 as a combination of hardware and software as known to those skilled in the art.

Program code to be loaded onto processor 310 or encoder/decoder 330 to perform the various aspects described in the present application may be stored in storage device 340
15 and subsequently loaded onto memory 320 for execution by processor 310. In accordance with various embodiments, one or more of processor 310, memory 320, storage device 340, and encoder/decoder module 330 may store one or more of various items during the performance of the processes described in the present application. Such stored items may include, but are not limited to, a point cloud frame, encoded/decoded
20 geometry/attributes videos/images or portions of the encoded/decoded geometry/attribute video/images, a bitstream, matrices, variables, and intermediate or final results from the processing of equations, formulas, operations, and operational logic.

In several embodiments, memory inside of the processor 310 and/or the encoder/decoder module 330 may be used to store instructions and to provide working memory for
25 processing that may be performed during encoding or decoding.

In other embodiments, however, a memory external to the processing device (for example, the processing device may be either the processor 310 or the encoder/decoder module 330) may be used for one or more of these functions. The external memory may be the memory 320 and/or the storage device 340, for example, a dynamic volatile memory
30 and/or a non-volatile flash memory. In several embodiments, an external non-volatile flash memory may be used to store the operating system of a television. In at least one embodiment, a fast external dynamic volatile memory such as a RAM may be used as working memory for video coding and decoding operations, such as for MPEG-2 part 2 (also known as ITU-T Recommendation H.262 and ISO/IEC 13818-2, also known as
35 MPEG-2 Video), HEVC (High Efficiency Video coding), VVC (Versatile Video Coding), or MPEG-I part 5 or part 9.

The input to the elements of system 300 may be provided through various input devices as indicated in block 390. Such input devices include, but are not limited to, (i) an RF
40 portion that may receive an RF signal transmitted, for example, over the air by a broadcaster, (ii) a Composite input terminal, (iii) a USB input terminal, and/or (iv) an HDMI input terminal.

In various embodiments, the input devices of block 390 may have associated respective

input processing elements as known in the art. For example, the RF portion may be associated with elements necessary for (i) selecting a desired frequency (also referred to as selecting a signal, or band-limiting a signal to a band of frequencies), (ii) down-converting the selected signal, (iii) band-limiting again to a narrower band of frequencies to select (for example) a signal frequency band which may be referred to as a channel in certain embodiments, (iv) demodulating the down-converted and band-limited signal, (v) performing error correction, and (vi) demultiplexing to select the desired stream of data packets. The RF portion of various embodiments may include one or more elements to perform these functions, for example, frequency selectors, signal selectors, band-limiters, channel selectors, filters, downconverters, demodulators, error correctors, and demultiplexers. The RF portion may include a tuner that performs various of these functions, including, for example, down-converting the received signal to a lower frequency (for example, an intermediate frequency or a near-baseband frequency) or to baseband.

In one set-top box embodiment, the RF portion and its associated input processing element may receive an RF signal transmitted over a wired (for example, cable) medium. Then, the RF portion may perform frequency selection by filtering, down-converting, and filtering again to a desired frequency band.

Various embodiments rearrange the order of the above-described (and other) elements, remove some of these elements, and/or add other elements performing similar or different functions.

Adding elements may include inserting elements in between existing elements, such as, for example, inserting amplifiers and an analog-to-digital converter. In various embodiments, the RF portion may include an antenna.

Additionally, the USB and/or HDMI terminals may include respective interface processors for connecting system 300 to other electronic devices across USB and/or HDMI connections. It is to be understood that various aspects of input processing, for example, Reed-Solomon error correction, may be implemented, for example, within a separate input processing IC or within processor 310 as necessary. Similarly, aspects of USB or HDMI interface processing may be implemented within separate interface ICs or within processor 310 as necessary. The demodulated, error corrected, and demultiplexed stream may be provided to various processing elements, including, for example, processor 310, and encoder/decoder 330 operating in combination with the memory and storage elements to process the data stream as necessary for presentation on an output device.

Various elements of system 300 may be provided within an integrated housing. Within the integrated housing, the various elements may be interconnected and transmit data therebetween using suitable connection arrangement 390, for example, an internal bus as known in the art, including the I2C bus, wiring, and printed circuit boards.

The system 300 may include communication interface 350 that enables communication with other devices via communication channel 700. The communication interface 350 may include, but is not limited to, a transceiver configured to transmit and to receive data over communication channel 700. The communication interface 350 may include, but is not

limited to, a modem or network card and the communication channel 700 may be implemented, for example, within a wired and/or a wireless medium.

5 Data may be streamed to the system 300, in various embodiments, using a Wi-Fi network such as IEEE 802.11. The Wi-Fi signal of these embodiments may be received over the communications channel 700 and the communications interface 350 which are adapted for Wi-Fi communications. The communications channel 700 of these embodiments may be typically connected to an access point or router that provides access to outside networks including the Internet for allowing streaming applications and other over-the-top communications.

10 Other embodiments may provide streamed data to the system 300 using a set-top box that delivers the data over the HDMI connection of the input block 390.

Still other embodiments may provide streamed data to the system 300 using the RF connection of the input block 390.

15 The streamed data may be used as a way for signaling information used by the system 300. The signaling information may comprise the bitstream B and/or information such as a number of points of a point cloud, coordinates or order $\alpha(P_1)$ of a first point in the 2D coordinates (ϕ, λ) system and/or sensor setup parameters such as such as an elementary azimuthal shift $\Delta\phi$ or an elevation angle θ_k associated with a sensor of the Lidar head 10.

20 It is to be appreciated that signaling may be accomplished in a variety of ways. For example, one or more syntax elements, flags, and so forth may be used to signal information to a corresponding decoder in various embodiments.

25 The system 300 may provide an output signal to various output devices, including a display 400, speakers 500, and other peripheral devices 600. The other peripheral devices 600 may include, in various examples of embodiments, one or more of a stand-alone DVR, a disk player, a stereo system, a lighting system, and other devices that provide a function based on the output of the system 300.

30 In various embodiments, control signals may be communicated between the system 300 and the display 400, speakers 500, or other peripheral devices 600 using signaling such as AV.Link (Audio/Video Link), CEC (Consumer Electronics Control), or other communications protocols that enable device-to-device control with or without user intervention.

The output devices may be communicatively coupled to system 300 via dedicated connections through respective interfaces 360, 370, and 380.

35 Alternatively, the output devices may be connected to system 300 using the communications channel 700 via the communications interface 350. The display 400 and speakers 500 may be integrated in a single unit with the other components of system 300 in an electronic device such as, for example, a television.

In various embodiments, the display interface 360 may include a display driver, such as, for example, a timing controller (T Con) chip.

40 The display 400 and speaker 500 may alternatively be separate from one or more of the

other components, for example, if the RF portion of input 390 is part of a separate set-top box. In various embodiments in which the display 400 and speakers 500 may be external components, the output signal may be provided via dedicated output connections, including, for example, HDMI ports, USB ports, or COMP outputs.

- 5 In **Fig. 1-21**, various methods are described herein, and each of the methods includes one or more steps or actions for achieving the described method. Unless a specific order of steps or actions is required for proper operation of the method, the order and/or use of specific steps and/or actions may be modified or combined.

- 10 Some examples are described with regard to block diagrams and/or operational flowcharts. Each block represents a circuit element, module, or portion of code which includes one or more executable instructions for implementing the specified logical function(s). It should also be noted that in other implementations, the function(s) noted in the blocks may occur out of the indicated order. For example, two blocks shown in succession may, in fact, be executed substantially concurrently or the blocks may
15 sometimes be executed in the reverse order, depending on the functionality involved.

- The implementations and aspects described herein may be implemented in, for example, a method or a process, an apparatus, a computer program, a data stream, a bitstream, or a signal. Even if only discussed in the context of a single form of implementation (for example, discussed only as a method), the implementation of features discussed may
20 also be implemented in other forms (for example, an apparatus or computer program).

- The methods may be implemented in, for example, a processor, which refers to processing devices in general, including, for example, a computer, a microprocessor, an integrated circuit, or a programmable logic device. Processors also include communication devices.

- 25 Additionally, the methods may be implemented by instructions being performed by a processor, and such instructions (and/or data values produced by an implementation) may be stored on a computer readable storage medium. A computer readable storage medium may take the form of a computer readable program product embodied in one or more computer readable medium(s) and having computer readable program code embodied
30 thereon that is executable by a computer. A computer readable storage medium as used herein may be considered a non-transitory storage medium given the inherent capability to store the information therein as well as the inherent capability to provide retrieval of the information therefrom. A computer readable storage medium may be, for example, but is not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor
35 system, apparatus, or device, or any suitable combination of the foregoing. It is to be appreciated that the following, while providing more specific examples of computer readable storage mediums to which the present embodiments may be applied, is merely an illustrative and not an exhaustive listing as is readily appreciated by one of ordinary skill in the art: a portable computer diskette; a hard disk; a read-only memory (ROM); an
40 erasable programmable read-only memory (EPROM or Flash memory); a portable compact disc read-only memory (CD-ROM); an optical storage device; a magnetic storage device; or any suitable combination of the foregoing.

The instructions may form an application program tangibly embodied on a processor-readable medium.

- Instructions may be, for example, in hardware, firmware, software, or a combination. Instructions may be found in, for example, an operating system, a separate application,
- 5 or a combination of the two. A processor may be characterized, therefore, as, for example, both a device configured to carry out a process and a device that includes a processor-readable medium (such as a storage device) having instructions for carrying out a process. Further, a processor-readable medium may store, in addition to or in lieu of instructions, data values produced by an implementation.
- 10 An apparatus may be implemented in, for example, appropriate hardware, software, and firmware. Examples of such apparatus include personal computers, laptops, smartphones, tablet computers, digital multimedia set top boxes, digital television receivers, personal video recording systems, connected home appliances, head mounted display devices (HMD, see-through glasses), projectors (beamers), "caves" (system including multiple
- 15 displays), servers, video encoders, video decoders, post-processors processing output from a video decoder, pre-processors providing input to a video encoder, web servers, set-top boxes, and any other device for processing a point cloud, a video or an image or other communication devices. As should be clear, the equipment may be mobile and even installed in a mobile vehicle.
- 20 Computer software may be implemented by the processor 310 or by hardware, or by a combination of hardware and software. As a non-limiting example, the embodiments may be also implemented by one or more integrated circuits. The memory 320 may be of any type appropriate to the technical environment and may be implemented using any appropriate data storage technology, such as optical memory devices, magnetic memory
- 25 devices, semiconductor-based memory devices, fixed memory, and removable memory, as non-limiting examples. The processor 310 may be of any type appropriate to the technical environment, and may encompass one or more of microprocessors, general purpose computers, special purpose computers, and processors based on a multi-core architecture, as non-limiting examples.
- 30 As will be evident to one of ordinary skill in the art, implementations may produce a variety of signals formatted to carry information that may be, for example, stored or transmitted. The information may include, for example, instructions for performing a method, or data produced by one of the described implementations. For example, a signal may be formatted to carry the bitstream of a described embodiment. Such a signal may be
- 35 formatted, for example, as an electromagnetic wave (for example, using a radio frequency portion of spectrum) or as a baseband signal. The formatting may include, for example, encoding a data stream and modulating a carrier with the encoded data stream. The information that the signal carries may be, for example, analog or digital information. The signal may be transmitted over a variety of different wired or wireless links, as is known.
- 40 The signal may be stored on a processor-readable medium.

The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting. As used herein, the singular forms "a", "an", and "the"

may be intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "includes/comprises" and/or "including/comprising" when used in this specification, may specify the presence of stated, for example, features, integers, steps, operations, elements, and/or components but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof. Moreover, when an element is referred to as being "responsive" or "connected" to another element, it may be directly responsive or connected to the other element, or intervening elements may be present. In contrast, when an element is referred to as being "directly responsive" or "directly connected" to other element, there are no intervening elements present.

It is to be appreciated that the use of any of the symbol/term "/", "and/or", and "at least one of", for example, in the cases of "A/B", "A and/or B" and "at least one of A and B", may be intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of both options (A and B). As a further example, in the cases of "A, B, and/or C" and "at least one of A, B, and C", such phrasing is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of the third listed option (C) only, or the selection of the first and the second listed options (A and B) only, or the selection of the first and third listed options (A and C) only, or the selection of the second and third listed options (B and C) only, or the selection of all three options (A and B and C). This may be extended, as is clear to one of ordinary skill in this and related arts, for as many items as are listed.

Various numeric values may be used in the present application. The specific values may be for example purposes and the aspects described are not limited to these specific values.

It will be understood that, although the terms first, second, etc. may be used herein to describe various elements, these elements are not 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 teachings of this application. No ordering is implied between a first element and a second element.

Reference to "one exemplary embodiment" or "an exemplary embodiment" or "one implementation" or "an implementation", as well as other variations thereof, is frequently used to convey that a particular feature, structure, characteristic, and so forth (described in connection with the embodiment/implementation) is included in at least one embodiment/implementation. Thus, the appearances of the phrase "in one exemplary embodiment" or "in an exemplary embodiment" or "in one implementation" or "in an implementation", as well as any other variations, appearing in various places throughout this application are not necessarily all referring to the same embodiment.

Similarly, reference herein to "in accordance with an exemplary embodiment/example/implementation" or "in an exemplary embodiment/example/implementation", as well as other variations thereof, is frequently

used to convey that a particular feature, structure, or characteristic (described in connection with the exemplary embodiment/example/implementation) may be included in at least one exemplary embodiment/example/implementation. Thus, the appearances of the expression “in accordance with an exemplary embodiment/example/implementation” or “in an exemplary embodiment/example/implementation” in various places in the specification are not necessarily all referring to the same exemplary embodiment/example/implementation, nor are separate or alternative exemplary embodiment/examples/implementation necessarily mutually exclusive of other exemplary embodiments/examples/implementation.

Reference numerals appearing in the claims are by way of illustration only and shall have no limiting effect on the scope of the claims. Although not explicitly described, the present embodiments/examples and variants may be employed in any combination or sub-combination.

When a figure is presented as a flow diagram, it should be understood that it also provides a block diagram of a corresponding apparatus. Similarly, when a figure is presented as a block diagram, it should be understood that it also provides a flow diagram of a corresponding method/process.

Although some of the diagrams include arrows on communication paths to show a primary direction of communication, it is to be understood that communication may occur in the opposite direction to the depicted arrows.

Various implementations involve decoding. “Decoding”, as used in this application, may encompass all or part of the processes performed, for example, on a received point cloud frame (including possibly a received bitstream which encodes one or more point cloud frames) in order to produce a final output suitable for display or for further processing in the reconstructed point cloud domain. In various embodiments, such processes include one or more of the processes typically performed by a decoder. In various embodiments, such processes also, or alternatively, include processes performed by a decoder of various implementations described in this application, for example,

As further examples, in one embodiment “decoding” may refer only to de-quantizing, in one embodiment “decoding” may refer to entropy decoding, in another embodiment “decoding” may refer only to differential decoding, and in another embodiment “decoding” may refer to combinations of de-quantizing, entropy decoding and differential decoding. Whether the phrase “decoding process” may be intended to refer specifically to a subset of operations or generally to the broader decoding process will be clear based on the context of the specific descriptions and is believed to be well understood by those skilled in the art.

Various implementations involve encoding. In an analogous way to the above discussion about “decoding”, “encoding” as used in this application may encompass all or part of the processes performed, for example, on an input point cloud frame in order to produce an encoded bitstream. In various embodiments, such processes include one or more of the processes typically performed by an encoder. In various embodiments, such processes also, or alternatively, include processes performed by an encoder of various

implementations described in this application.

As further examples, in one embodiment “encoding” may refer only to quantizing, in one embodiment “encoding” may refer only to entropy encoding, in another embodiment “encoding” may refer only to differential encoding, and in another embodiment “encoding” may refer to combinations of quantizing, differential encoding and entropy encoding. Whether the phrase “encoding process” may be intended to refer specifically to a subset of operations or generally to the broader encoding process will be clear based on the context of the specific descriptions and is believed to be well understood by those skilled in the art.

Additionally, this application may refer to “determining” various pieces of information. Determining the information may include one or more of, for example, estimating the information, calculating the information, predicting the information, or retrieving the information from memory.

Further, this application may refer to “accessing” various pieces of information. Accessing the information may include one or more of, for example, receiving the information, retrieving the information (for example, from memory), storing the information, moving the information, copying the information, calculating the information, determining the information, predicting the information, or estimating the information.

Additionally, this application may refer to “receiving” various pieces of information. Receiving is, as with “accessing”, intended to be a broad term. Receiving the information may include one or more of, for example, accessing the information, or retrieving the information (for example, from memory). Further, “receiving” is typically involved, in one way or another, during operations such as, for example, storing the information, processing the information, transmitting the information, moving the information, copying the information, erasing the information, calculating the information, determining the information, predicting the information, or estimating the information.

Also, as used herein, the word “signal” refers to, among other things, indicating something to a corresponding decoder. For example, in certain embodiments the encoder signals a particular information such as at least one binary data f_j , the residual R , a number of points of the point cloud or coordinates or order $o(P_1)$ of a first point in the 2D coordinates (ϕ, λ) system or sensor setup parameters such as the elementary azimuthal shift $\Delta\phi$ or an elevation angle θ_k associated with a sensor $\mathbf{k}$. In this way, in an embodiment the same parameter may be used at both the encoder side and the decoder side. Thus, for example, an encoder may transmit (explicit signaling) a particular parameter to the decoder so that the decoder may use the same particular parameter. Conversely, if the decoder already has the particular parameter as well as others, then signaling may be used without transmitting (implicit signaling) to simply allow the decoder to know and select the particular parameter. By avoiding transmission of any actual functions, a bit savings is realized in various embodiments. It is to be appreciated that signaling may be accomplished in a variety of ways. For example, one or more syntax elements, flags, and so forth are used to signal information to a corresponding decoder in various embodiments. While the preceding relates to the verb form of the word “signal”, the word “signal” may

also be used herein as a noun.

A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made. For example, elements of different implementations may be combined, supplemented, modified, or removed to produce other
5 implementations. Additionally, one of ordinary skill will understand that other structures and processes may be substituted for those disclosed and the resulting implementations will perform at least substantially the same function(s), in at least substantially the same way(s), to achieve at least substantially the same result(s) as the implementations disclosed. Accordingly, these and other implementations are contemplated by this
10 application.

CLAIMS

1. A method of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object, points of the point cloud being ordered based on azimuthal angles (ϕ_n) representing capture angles of sensors and sensor indices (λ_n) associated with sensors, wherein the method comprises encoding, into a bitstream, at least one order index difference (Δo_n) representing a difference between order indices of two consecutive ordered points by:
 - obtaining (810) at least one binary data (f_j) by binarizing the at least one order index difference (Δo_n); and
 - for each binary data, selecting (820) a context based on a distance (C_j) between an azimuthal angle (ϕ^j) associated with the binary data and an azimuthal angle (ϕ_{penult}, ϕ_{al}) of an already encoded point, and context-based entropy coding (830) the binary data in the bitstream, based on the selected context.
2. The method of claim 1, wherein the already encoded point is a penultimate already coded point (P_{penult}) with a same sensor index as a sensor index (λ^j) associated with the binary data (f_j).
3. A method of decoding a point cloud from a bitstream of encoded point cloud data representing a physical object, wherein the method comprises decoding at least one order index difference (Δo_n) representing a difference between order indices of two consecutive ordered points based on at least one binary data (f_j) decoded from the bitstream, each binary data (f_j) being decoded by:
 - selecting (910) a context based on a distance (C_j) between an azimuthal angle (ϕ^j) associated with the binary data (f_j) and an azimuthal angle (ϕ_{penult}, ϕ_{al}) of an already decoded point;
 - context-based entropy decoding (920) said at least one binary data based on the selected context and a probability information relative to the binary data decoded from the bitstream; and
 - decoding (930) an order index difference (Δo_n) from the at least one context-based entropy decoded binary data (f_j).
4. The method of claim 3, wherein the already decoded point is a penultimate already decoded point (P_{penult}) with a same sensor index as a sensor index (λ^j) associated with the binary data (f_j).
5. The method of claim 3, wherein a context for decoding a binary data being selected from a context table indexed by a context index (ctxldx), and wherein the context index for a binary data (f_j) equals the distance (C_j) between the azimuthal angle (ϕ^j) associated with the binary data (f_j) and an azimuthal angle (ϕ_{penult}, ϕ_{al}) of the already decoded point.
6. The method of claim 3, wherein a context for decoding a binary data being selected from a context table indexed by a context index (ctxldx), and wherein the context index (ctxldx) for a binary data (f_j) depends on the rank (j) of said binary data in a series of binary data representing the order index difference (Δo_n).
7. The method of claim 3, wherein a context for decoding a binary data being selected from a context table indexed by a context index (ctxldx), and wherein the context index (ctxldx) for a binary data (f_j) depends on a sensor index (λ^j) associated with the binary data (f_j).

8. The method of claim 3, a context for decoding a binary data being selected from a context table indexed by a context index (ctxldx), and the context index (ctxldx) for a binary data (f_i) depends on sensor packs regrouping sensors with close elevation angles.
9. An apparatus of encoding of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object, points of the point cloud being ordered based on azimuthal angles representing capture angles of sensors and sensor indices associated with sensors, wherein the apparatus comprises one or more processors configured to encode, into a bitstream, at least one order index difference representing a difference between order indices of two consecutive ordered points by:
- 10 obtaining at least one binary data by binarizing the at least one order index difference; and
- for each binary data, selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already encoded point, and context-based entropy coding the binary data in the bitstream, based on the selected context.
10. An apparatus of decoding, from a bitstream, a point of a point cloud representing a physical object, wherein the apparatus comprises one or more processors configured to decode at least one order index difference representing a difference between order indices of two consecutive ordered points based on at least one binary data decoded from the bitstream, each binary data being decoded by:
- selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already decoded point;
- context-based entropy decoding said at least one binary data based on the selected context and a probability information relative to the binary data decoded from the bitstream; and
- decoding an order index difference from the at least one context-based entropy decoded binary data.
11. A bitstream of encoded point cloud data representing a physical object, wherein the bitstream comprises probability information relative to at least one binary data representing an order index difference representing a difference between order indices of two consecutive ordered points of the point cloud.
12. A computer program product including instructions which, when the program is executed by one or more processors, causes the one or more processors to carry out a method of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object, points of the point cloud being ordered based on azimuthal angles representing capture angles of sensors and sensor indices associated with sensors, the method comprising encoding, into a bitstream, at least one order index difference representing a difference between order indices of two consecutive ordered points by:
- 40 obtaining at least one binary data by binarizing the at least one order index difference; and
- for each binary data, selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already

encoded point, and context-based entropy coding the binary data in the bitstream, based on the selected context.

13. A non-transitory storage medium carrying instructions of program code for executing a method of encoding a point cloud into a bitstream of encoded point cloud data representing a physical object, points of the point cloud being ordered based on azimuthal angles representing capture angles of sensors and sensor indices associated with sensors, the method comprising encoding, into a bitstream, at least one order index difference representing a difference between order indices of two consecutive ordered points by:
- 5 obtaining at least one binary data by binarizing the at least one order index difference; and
- for each binary data, selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already encoded point, and context-based entropy coding the binary data in the bitstream, based on the selected context.
14. A computer program product including instructions which, when the program is executed by one or more processors, causes the one or more processors to carry out a method of decoding, from a bitstream, a point of a point cloud representing a physical object, the method comprises decoding at least one order index difference representing a difference between order indices of two consecutive ordered points based on at least one binary data decoded from the bitstream, each binary data being decoded by:
- 20 selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already decoded point;
- context-based entropy decoding said at least one binary data based on the selected context and a probability information relative to the binary data decoded from the bitstream; and
- 25 decoding an order index difference from the at least one context-based entropy decoded binary data.
15. A non-transitory storage medium carrying instructions of program code for executing a method of decoding, from a bitstream, a point of a point cloud representing a physical object, the method comprises decoding at least one order index difference representing a difference between order indices of two consecutive ordered points based on at least one binary data decoded from the bitstream, each binary data being decoded by:
- 30 selecting a context based on a distance between an azimuthal angle associated with the binary data and an azimuthal angle of an already decoded point;
- context-based entropy decoding said at least one binary data based on the selected context and a probability information relative to the binary data decoded from the bitstream; and
- 35 decoding an order index difference from the at least one context-based entropy decoded binary data.
- 40