Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 3090919 A1 2019/08/22

(21) 3 090 919

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2019/02/13
(87) Date publication PCT/PCT Publication Date: 2019/08/22
(85) Entrée phase nationale/National Entry: 2020/08/10
(86) N° demande PCT/PCT Application No.: JP 2019/005063
(87) N° publication PCT/PCT Publication No.: 2019/159956
(30) Priorité/Priority: 2018/02/14 (US62/630,514)

(51) Cl.Int./Int.Cl. *G06T 9/40* (2006.01), *G06T 19/00* (2011.01), *H04N 19/96* (2014.01)

(71) Demandeur/Applicant:
PANASONIC INTELLECTUAL PROPERTY CORPORATION OF AMERICA, US

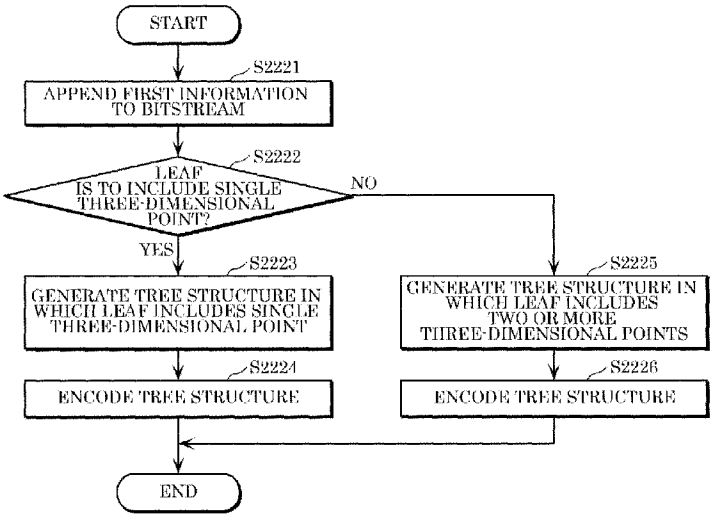
(72) Inventeur/Inventor:
SUGIO, TOSHIYASU, JP

(74) Agent: OSLER, HOSKIN & HARCOURT LLP

(54) Titre : PROCEDE DE CODAGE DE DONNEES TRIDIMENSIONNELLES, PROCEDE DE DECODAGE DE DONNEES TRIDIMENSIONNELLES, DISPOSITIF DE CODAGE DE DONNEES TRIDIMENSIONNELLES ET DISPOSITIF DE DECODAGE DE DONNEES TRIDIMENSIONNELLE

(54) Title: THREE-DIMENSIONAL DATA ENCODING METHOD, THREE-DIMENSIONAL DATA DECODING METHOD, THREE-DIMENSIONAL DATA ENCODING DEVICE, AND THREE-DIMENSIONAL DATA DECODING DEVICE

FIG. 161



(57) Abrégé/Abstract:
This three-dimensional data encoding method includes: adding, to a bitstream, first information indicating whether a leaf included in an N-branched tree structure (N is an integer of 2 or greater) having multiple three-dimensional points included in three-dimensional data, includes one three-dimensional point or includes two or more three-dimensional points (S2221); and when the first information indicates that the leaf includes one three-dimensional point (Yes in S2222), generating an N-branched tree structure such that the leaf includes one three-dimensional point (S2223) and encoding the N-branched tree structure (S2224), whereas when the first information indicates that the leaf includes two or more three-dimensional points (No in S2222), generating an N-branched tree structure such that the leaf includes two or more three-dimensional points (S2225) and encoding the N-branched tree structure (S2226).

ABSTRACT

A three-dimensional data encoding method includes: appending, to a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2 (S2221); when the first information indicates that the leaf is to include a single three-dimensional point (YES in S2222), generating an N-ary tree structure in which a leaf includes a single three-dimensional point (S2223), and encoding the N-ary tree structure (S2224); and when the first information indicates that the leaf is to include two or more three-dimensional points (NO in S2222), generating an N-ary tree structure in which a leaf includes two or more three-dimensional points (S2225), and encoding the N-ary tree structure (S2226).

DESCRIPTION

THREE-DIMENSIONAL DATA ENCODING METHOD, THREE-DIMENSIONAL DATA DECODING METHOD, THREE-DIMENSIONAL DATA ENCODING DEVICE, AND THREE-DIMENSIONAL DATA DECODING DEVICE

TECHNICAL FIELD

[0001]

The present disclosure relates to a three-dimensional data encoding method, a three-dimensional data decoding method, a three-dimensional data encoding device, and a three-dimensional data decoding device.

BACKGROUND ART

[0002]

Devices or services utilizing three-dimensional data are expected to find their widespread use in a wide range of fields, such as computer vision that enables autonomous operations of cars or robots, map information, monitoring, infrastructure inspection, and video distribution. Three-dimensional data is obtained through various means including a distance sensor such as a rangefinder, as well as a stereo camera and a combination of a plurality of monocular cameras.

[0003]

Methods of representing three-dimensional data include a method known as a point cloud scheme that represents the shape of a three-dimensional structure by a point group in a three-dimensional space. In the point cloud scheme, the positions and colors of a point group are stored. While point cloud is expected to be a mainstream method of representing three-dimensional data, a massive amount of data of a point group necessitates compression of the amount of three-dimensional data by encoding for accumulation and transmission, as in the case of a two-dimensional moving picture (examples include MPEG-4 AVC and HEVC

standardized by MPEG).

[0004]

Meanwhile, point cloud compression is partially supported by, for example, an open-source library (Point Cloud Library) for point cloud-related processing.

[0005]

Furthermore, a technique for searching for and displaying a facility located in the surroundings of the vehicle is known (for example, see Patent Literature (PTL) 1).

Citation List

Patent Literature

[0006]

PTL 1: International Publication WO 2014/020663

SUMMARY OF THE INVENTION

TECHNICAL PROBLEM

[0007]

There have been a demand for improving coding efficiency in encoding of three-dimensional data, and a demand for reducing a processing amount.

[0008]

The present disclosure has an object to provide a three-dimensional data encoding method, a three-dimensional data decoding method, a three-dimensional data encoding device, or a three-dimensional data decoding device that is capable of both improving the coding efficiency and reducing the processing amount.

SOLUTION TO PROBLEM

[0009]

A three-dimensional data encoding method according to one aspect of the present disclosure includes: appending, to a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more

three-dimensional points, where N is an integer greater than or equal to 2; when the first information indicates that the leaf is to include a single three-dimensional point, generating an N-ary tree structure in which a leaf includes a single three-dimensional point, and encoding the N-ary tree structure; and when the first information indicates that the leaf is to include two or more three-dimensional points, generating an N-ary tree structure in which a leaf includes two or more three-dimensional points, and encoding the N-ary tree structure.

[0010]

A three-dimensional data decoding method according to one aspect of the present disclosure includes: decoding, from a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2; when the first information indicates that the leaf is to include a single three-dimensional point, decoding an N-ary tree structure in which a leaf includes a single three-dimensional point; and when the first information indicates that the leaf is to include two or more three-dimensional points, decoding an N-ary tree structure in which a leaf includes two or more three-dimensional points.

ADVANTAGEOUS EFFECT OF INVENTION

[0011]

The present disclosure provides a three-dimensional data encoding method, a three-dimensional data decoding method, a three-dimensional data encoding device, or a three-dimensional data decoding device that is capable of both improving coding efficiency and reducing a processing amount.

BRIEF DESCRIPTION OF DRAWINGS

[0012]

FIG. 1 is a diagram showing the structure of encoded three-dimensional data according to Embodiment 1.

FIG. 2 is a diagram showing an example of prediction structures among SPCs that belong to the lowermost layer in a GOS according to Embodiment 1.

FIG. 3 is a diagram showing an example of prediction structures among layers according to Embodiment 1.

FIG. 4 is a diagram showing an example order of encoding GOSs according to Embodiment 1.

FIG. 5 is a diagram showing an example order of encoding GOSs according to Embodiment 1.

FIG. 6 is a block diagram of a three-dimensional data encoding device according to Embodiment 1.

FIG. 7 is a flowchart of encoding processes according to Embodiment 1.

FIG. 8 is a block diagram of a three-dimensional data decoding device according to Embodiment 1.

FIG. 9 is a flowchart of decoding processes according to Embodiment 1.

FIG. 10 is a diagram showing an example of meta information according to Embodiment 1.

FIG. 11 is a diagram showing an example structure of a SWLD according to Embodiment 2.

FIG. 12 is a diagram showing example operations performed by a server and a client according to Embodiment 2.

FIG. 13 is a diagram showing example operations performed by the server and a client according to Embodiment 2.

FIG. 14 is a diagram showing example operations performed by the server and the clients according to Embodiment 2.

FIG. 15 is a diagram showing example operations performed by the server and the clients according to Embodiment 2.

FIG. 16 is a block diagram of a three-dimensional data encoding device according to

Embodiment 2.

FIG. 17 is a flowchart of encoding processes according to Embodiment 2.

FIG. 18 is a block diagram of a three-dimensional data decoding device according to Embodiment 2.

FIG. 19 is a flowchart of decoding processes according to Embodiment 2.

FIG. 20 is a diagram showing an example structure of a WLD according to Embodiment 2.

FIG. 21 is a diagram showing an example octree structure of the WLD according to Embodiment 2.

FIG. 22 is a diagram showing an example structure of a SWLD according to Embodiment 2.

FIG. 23 is a diagram showing an example octree structure of the SWLD according to Embodiment 2.

FIG. 24 is a block diagram of a three-dimensional data creation device according to Embodiment 3.

FIG. 25 is a block diagram of a three-dimensional data transmission device according to Embodiment 3.

FIG. 26 is a block diagram of a three-dimensional information processing device according to Embodiment 4.

FIG. 27 is a block diagram of a three-dimensional data creation device according to Embodiment 5.

FIG. 28 is a diagram showing a structure of a system according to Embodiment 6.

FIG. 29 is a block diagram of a client device according to Embodiment 6.

FIG. 30 is a block diagram of a server according to Embodiment 6.

FIG. 31 is a flowchart of a three-dimensional data creation process performed by the client device according to Embodiment 6.

FIG. 32 is a flowchart of a sensor information transmission process performed by the

client device according to Embodiment 6.

FIG. 33 is a flowchart of a three-dimensional data creation process performed by the server according to Embodiment 6.

FIG. 34 is a flowchart of a three-dimensional map transmission process performed by the server according to Embodiment 6.

FIG. 35 is a diagram showing a structure of a variation of the system according to Embodiment 6.

FIG. 36 is a diagram showing a structure of the server and client devices according to Embodiment 6.

FIG. 37 is a block diagram of a three-dimensional data encoding device according to Embodiment 7.

FIG. 38 is a diagram showing an example of a prediction residual according to Embodiment 7.

FIG. 39 is a diagram showing an example of a volume according to Embodiment 7.

FIG. 40 is a diagram showing an example of an octree representation of the volume according to Embodiment 7.

FIG. 41 is a diagram showing an example of bit sequences of the volume according to Embodiment 7.

FIG. 42 is a diagram showing an example of an octree representation of a volume according to Embodiment 7.

FIG. 43 is a diagram showing an example of the volume according to Embodiment 7.

FIG. 44 is a diagram for describing an intra prediction process according to Embodiment 7.

FIG. 45 is a diagram for describing a rotation and translation process according to Embodiment 7.

FIG. 46 is a diagram showing an example syntax of an RT flag and RT information

according to Embodiment 7.

FIG. 47 is a diagram for describing an inter prediction process according to Embodiment 7.

FIG. 48 is a block diagram of a three-dimensional data decoding device according to Embodiment 7.

FIG. 49 is a flowchart of a three-dimensional data encoding process performed by the three-dimensional data encoding device according to Embodiment 7.

FIG. 50 is a flowchart of a three-dimensional data decoding process performed by the three-dimensional data decoding device according to Embodiment 7.

FIG. 51 is a diagram showing a structure of a distribution system according to Embodiment 8.

FIG. 52 is a diagram showing an example structure of a bitstream of an encoded three-dimensional map according to Embodiment 8.

FIG. 53 is a diagram for describing an advantageous effect on encoding efficiency according to Embodiment 8.

FIG. 54 is a flowchart of processes performed by a server according to Embodiment 8.

FIG. 55 is a flowchart of processes performed by a client according to Embodiment 8.

FIG. 56 is a diagram showing an example syntax of a submap according to Embodiment 8.

FIG. 57 is a diagram schematically showing a switching process of an encoding type according to Embodiment 8.

FIG. 58 is a diagram showing an example syntax of a submap according to Embodiment 8.

FIG. 59 is a flowchart of a three-dimensional data encoding process according to Embodiment 8.

FIG. 60 is a flowchart of a three-dimensional data decoding process according to Embodiment 8.

FIG. 61 is a diagram schematically showing an operation of a variation of the switching process of the encoding type according to Embodiment 8.

FIG. 62 is a diagram schematically showing an operation of a variation of the switching process of the encoding type according to Embodiment 8.

FIG. 63 is a diagram schematically showing an operation of a variation of the switching process of the encoding type according to Embodiment 8.

FIG. 64 is a diagram schematically showing an operation of a variation of a calculation process of a differential value according to Embodiment 8.

FIG. 65 is a diagram schematically showing an operation of a variation of the calculation process of the differential value according to Embodiment 8.

FIG. 66 is a diagram schematically showing an operation of a variation of the calculation process of the differential value according to Embodiment 8.

FIG. 67 is a diagram schematically showing an operation of a variation of the calculation process of the differential value according to Embodiment 8.

FIG. 68 is a diagram showing an example syntax of a volume according to Embodiment 8.

FIG. 69 is a diagram showing an example of an important area according to Embodiment 9.

FIG. 70 is a diagram showing an example of an occupancy code according to Embodiment 9.

FIG. 71 is a diagram showing an example of a quadtree structure according to Embodiment 9.

FIG. 72 is a diagram showing an example of an occupancy code and a location code according to Embodiment 9.

FIG. 73 is a diagram showing an example of three-dimensional points obtained

through LiDAR according to Embodiment 9.

FIG. 74 is a diagram showing an example of an octree structure according to Embodiment 9.

FIG. 75 is a diagram showing an example of hybrid encoding according to Embodiment 9.

FIG. 76 is a diagram for describing a method for switching between location encoding and occupancy encoding according to Embodiment 9.

FIG. 77 is a diagram showing an example of a location encoded bitstream according to Embodiment 9.

FIG. 78 is a diagram showing an example of a hybrid encoded bitstream according to Embodiment 9.

FIG. 79 is a diagram showing an occupancy code tree structure of important three-dimensional points according to Embodiment 9.

FIG. 80 is a diagram showing an occupancy code tree structure of non-important three-dimensional points according to Embodiment 9.

FIG. 81 is a diagram showing an example of a hybrid encoded bitstream according to Embodiment 9.

FIG. 82 is a diagram showing an example of a bitstream including encoding mode information according to Embodiment 9.

FIG. 83 is a diagram showing an example syntax according to Embodiment 9.

FIG. 84 is a flowchart of an encoding process according to Embodiment 9.

FIG. 85 is a flowchart of a node encoding process according to Embodiment 9.

FIG. 86 is a flowchart of a decoding process according to Embodiment 9.

FIG. 87 is a flowchart of a node decoding process according to Embodiment 9.

FIG. 88 is a diagram illustrating an example of a tree structure according to Embodiment 10.

FIG. 89 is a graph showing an example of the number of valid leaves of each branch

according to Embodiment 10.

FIG. 90 is a diagram illustrating an application example of encoding schemes according to Embodiment 10.

FIG. 91 is a diagram illustrating an example of a dense branch area according to Embodiment 10.

FIG. 92 is a diagram illustrating an example of a dense three-dimensional point cloud according to Embodiment 10.

FIG. 93 is a diagram illustrating an example of a sparse three-dimensional point cloud according to Embodiment 10.

FIG. 94 is a flowchart of an encoding process according to Embodiment 10.

FIG. 95 is a flowchart of a decoding process according to Embodiment 10.

FIG. 96 is a flowchart of an encoding process according to Embodiment 10.

FIG. 97 is a flowchart of a decoding process according to Embodiment 10.

FIG. 98 is a flowchart of an encoding process according to Embodiment 10.

FIG. 99 is a flowchart of a decoding process according to Embodiment 10.

FIG. 100 is a flowchart of a process of separating three-dimensional points according to Embodiment 10.

FIG. 101 is a diagram illustrating an example of a syntax according to Embodiment 10.

FIG. 102 is a diagram illustrating an example of a dense branch according to Embodiment 10.

FIG. 103 is a diagram illustrating an example of a sparse branch according to Embodiment 10.

FIG. 104 is a flowchart of an encoding process according to a variation of Embodiment 10.

FIG. 105 is a flowchart of a decoding process according to the variation of Embodiment 10.

FIG. 106 is a flowchart of a process of separating three-dimensional points according to the variation of Embodiment 10.

FIG. 107 is a diagram illustrating an example of a syntax according to the variation of Embodiment 10.

FIG. 108 is a flowchart of an encoding process according to Embodiment 10.

FIG. 109 is a flowchart of a decoding process according to Embodiment 10.

FIG. 110 is a diagram illustrating an example of a tree structure according to Embodiment 11.

FIG. 111 is a diagram illustrating an example of occupancy codes according to Embodiment 11.

FIG. 112 is a diagram schematically illustrating an operation performed by a three-dimensional data encoding device according to Embodiment 11.

FIG. 113 is a diagram illustrating an example of geometry information according to Embodiment 11.

FIG. 114 is a diagram illustrating an example of selecting a coding table using geometry information according to Embodiment 11.

FIG. 115 is a diagram illustrating an example of selecting a coding table using structure information according to Embodiment 11.

FIG. 116 is a diagram illustrating an example of selecting a coding table using attribute information according to Embodiment 11.

FIG. 117 is a diagram illustrating an example of selecting a coding table using attribute information according to Embodiment 11.

FIG. 118 is a diagram illustrating an example of a structure of a bitstream according to Embodiment 11.

FIG. 119 is a diagram illustrating an example of a coding table according to Embodiment 11.

FIG. 120 is a diagram illustrating an example of a coding table according to

Embodiment 11.

FIG. 121 is a diagram illustrating an example of a structure of a bitstream according to Embodiment 11.

FIG. 122 is a diagram illustrating an example of a coding table according to Embodiment 11.

FIG. 123 is a diagram illustrating an example of a coding table according to Embodiment 11.

FIG. 124 is a diagram illustrating an example of bit numbers of an occupancy code according to Embodiment 11.

FIG. 125 is a flowchart of an encoding process using geometry information according to Embodiment 11.

FIG. 126 is a flowchart of a decoding process using geometry information according to Embodiment 11.

FIG. 127 is a flowchart of an encoding process using structure information according to Embodiment 11.

FIG. 128 is a flowchart of a decoding process using structure information according to Embodiment 11.

FIG. 129 is a flowchart of an encoding process using attribute information according to Embodiment 11.

FIG. 130 is a flowchart of a decoding process using attribute information according to Embodiment 11.

FIG. 131 is a flowchart of a process of selecting a coding table using geometry information according to Embodiment 11.

FIG. 132 is a flowchart of a process of selecting a coding table using structure information according to Embodiment 11.

FIG. 133 is a flowchart of a process of selecting a coding table using attribute information according to Embodiment 11.

FIG. 134 is a block diagram of a three-dimensional data encoding device according to Embodiment 11.

FIG. 135 is a block diagram of a three-dimensional data decoding device according to Embodiment 11.

FIG. 136 is a diagram illustrating a reference relationship in an octree structure according to Embodiment 12.

FIG. 137 is a diagram illustrating a reference relationship in a spatial region according to Embodiment 12.

FIG. 138 is a diagram illustrating an example of neighboring reference nodes according to Embodiment 12.

FIG. 139 is a diagram illustrating a relationship between a parent node and nodes according to Embodiment 12.

FIG. 140 is a diagram illustrating an example of an occupancy code of the parent node according to Embodiment 12.

FIG. 141 is a block diagram of a three-dimensional data encoding device according to Embodiment 12.

FIG. 142 is a block diagram of a three-dimensional data decoding device according to Embodiment 12.

FIG. 143 is a flowchart of a three-dimensional data encoding process according to Embodiment 12.

FIG. 144 is a flowchart of a three-dimensional data decoding process according to Embodiment 12.

FIG. 145 is a diagram illustrating an example of selecting a coding table according to Embodiment 12.

FIG. 146 is a diagram illustrating a reference relationship in a spatial region according to Variation 1 of Embodiment 12.

FIG. 147 is a diagram illustrating an example of a syntax of header information

according to Variation 1 of Embodiment 12.

FIG. 148 is a diagram illustrating an example of a syntax of header information according to Variation 1 of Embodiment 12.

FIG. 149 is a diagram illustrating an example of neighboring reference nodes according to Variation 2 of Embodiment 12.

FIG. 150 is a diagram illustrating an example of a current node and neighboring nodes according to Variation 2 of Embodiment 12.

FIG. 151 is a diagram illustrating a reference relationship in an octree structure according to Variation 3 of Embodiment 12.

FIG. 152 is a diagram illustrating a reference relationship in a spatial region according to Variation 3 of Embodiment 12.

FIG. 153 is a diagram illustrating an example of a syntax of header information according to Embodiment 13.

FIG. 154 is a diagram illustrating a configuration example of an octree when mode information according to Embodiment 13 indicates 1.

FIG. 155 is a diagram illustrating a configuration example of an octree when mode information according to Embodiment 13 indicates 0.

FIG. 156 is a diagram illustrating an example of a syntax of information of a node according to Embodiment 13.

FIG. 157 is a flowchart of a three-dimensional data encoding process according to Embodiment 13.

FIG. 158 is a flowchart of a three-dimensional data decoding process according to Embodiment 13.

FIG. 159 is a block diagram of a three-dimensional data encoding device according to Embodiment 13.

FIG. 160 is a block diagram of a three-dimensional data decoding device according to Embodiment 13.

FIG. 161 is a flowchart of a three-dimensional data encoding process according to Embodiment 13.

FIG. 162 is a flowchart of a three-dimensional data decoding process according to Embodiment 13.

DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0013]

A three-dimensional data encoding method according to one aspect of the present disclosure includes: appending, to a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2; when the first information indicates that the leaf is to include a single three-dimensional point, generating an N-ary tree structure in which a leaf includes a single three-dimensional point, and encoding the N-ary tree structure; and when the first information indicates that the leaf is to include two or more three-dimensional points, generating an N-ary tree structure in which a leaf includes two or more three-dimensional points, and encoding the N-ary tree structure.

[0014]

With this, the three-dimensional data encoding method makes it possible to selectively use the tree structure in which the leaf includes a single three-dimensional point, and the tree structure in which the leaf includes two or more three-dimensional points. Accordingly, the three-dimensional data encoding method makes it possible to improve the coding efficiency.

[0015]

For example, the three-dimensional data encoding method may further include appending second information about the leaf to the bitstream when the first information indicates that the leaf is to include two or more three-dimensional points. When the first information indicates that the leaf is to include a single three-dimensional point, no second

information may be appended to the bitstream.

[0016]

With this, the three-dimensional data encoding method makes it possible to improve the coding efficiency by appending no second information to the bitstream when the leaf is to include a single three-dimensional point.

[0017]

For example, the second information may indicate a total number of three-dimensional points included in the leaf.

[0018]

For example, the first information may indicate whether each of leaves to be included in the N-ary tree structure is to include a single three-dimensional point or two or more three-dimensional points. When the first information indicates that each of the leaves is to include a single three-dimensional point, an N-ary tree structure may be generated in which each of leaves includes a single three-dimensional point, and the N-ary tree structure may be encoded. When the first information indicates that each of the leaves is to include two or more three-dimensional points, an N-ary tree structure may be generated in which each of leaves includes two or more three-dimensional points, and the N-ary tree structure may be encoded.

[0019]

With this, since it is possible to control the formats of the leaves using the single first information, the three-dimensional data encoding method makes it possible to improve the coding efficiency.

[0020]

For example, the two or more three-dimensional points included in the leaf may have mutually different space coordinates.

[0021]

For example, the two or more three-dimensional points included in the leaf may

have same space coordinates and mutually different attribute information.

[0022]

For example, each of the two or more three-dimensional points included in the leaf may have coordinate information and attribute information.

[0023]

A three-dimensional data decoding method according to one aspect of the present disclosure includes: decoding, from a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2; when the first information indicates that the leaf is to include a single three-dimensional point, decoding an N-ary tree structure in which a leaf includes a single three-dimensional point; and when the first information indicates that the leaf is to include two or more three-dimensional points, decoding an N-ary tree structure in which a leaf includes two or more three-dimensional points.

[0024]

With this, the three-dimensional data decoding method makes it possible to selectively use the tree structure in which the leaf includes a single three-dimensional point, and the tree structure in which the leaf includes two or more three-dimensional points. Accordingly, the three-dimensional data decoding method makes it possible to improve the coding efficiency.

[0025]

For example, the three-dimensional data decoding method may further include decoding, from the bitstream, second information about the leaf when the first information indicates that the leaf is to include two or more three-dimensional points. When the first information indicates that the leaf is to include a single three-dimensional point, no second information may be appended to the bitstream.

[0026]

With this, since the second information need not be appended to the bitstream when the leaf is to include a single three-dimensional point, it is possible to improve the coding efficiency.

[0027]

For example, the second information may indicate a total number of three-dimensional points included in the leaf.

[0028]

For example, the first information may indicate whether each of leaves to be included in the N-ary tree structure is to include a single three-dimensional point or two or more three-dimensional points. When the first information indicates that each of the leaves is to include a single three-dimensional point, an N-ary tree structure may be decoded in which each of leaves includes a single three-dimensional point. When the first information indicates that each of the leaves is to include two or more three-dimensional points, an N-ary tree structure may be decoded in which each of leaves includes two or more three-dimensional points.

[0029]

With this, since it is possible to control the formats of the leaves using the single first information, the three-dimensional data decoding method makes it possible to improve the coding efficiency.

[0030]

For example, the two or more three-dimensional points included in the leaf may have mutually different space coordinates.

[0031]

For example, the two or more three-dimensional points included in the leaf may have same space coordinates and mutually different attribute information.

[0032]

For example, each of the two or more three-dimensional points included in the leaf may have coordinate information and attribute information.

[0033]

A three-dimensional data encoding device according to one aspect of the present disclosure includes a processor and memory. Using the memory, the processor: appends, to a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2; when the first information indicates that the leaf is to include a single three-dimensional point, generates an N-ary tree structure in which a leaf includes a single three-dimensional point, and encodes the N-ary tree structure; and when the first information indicates that the leaf is to include two or more three-dimensional points, generates an N-ary tree structure in which a leaf includes two or more three-dimensional points, and encodes the N-ary tree structure.

[0034]

With this, the three-dimensional data encoding device can selectively use the tree structure in which the leaf includes a single three-dimensional point, and the tree structure in which the leaf includes two or more three-dimensional points. Accordingly, the three-dimensional data encoding device can improve the coding efficiency.

[0035]

A three-dimensional data decoding device according to one aspect of the present disclosure includes a processor and memory. Using the memory, the processor: decodes, from a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2; when the first information indicates that the leaf is to include a single three-dimensional point, decodes an N-ary tree structure in which a leaf includes a

single three-dimensional point; and when the first information indicates that the leaf is to include two or more three-dimensional points, decodes an N-ary tree structure in which a leaf includes two or more three-dimensional points.

[0036]

With this, the three-dimensional data decoding device can selectively use the tree structure in which the leaf includes a single three-dimensional point, and the tree structure in which the leaf includes two or more three-dimensional points. Accordingly, the three-dimensional data decoding device can improve the coding efficiency.

[0037]

Note that these general or specific aspects may be implemented as a system, a method, an integrated circuit, a computer program, or a computer-readable recording medium such as a CD-ROM, or may be implemented as any combination of a system, a method, an integrated circuit, a computer program, and a recording medium.

[0038]

The following describes embodiments with reference to the drawings. Note that the following embodiments show exemplary embodiments of the present disclosure. The numerical values, shapes, materials, structural components, the arrangement and connection of the structural components, steps, the processing order of the steps, etc. shown in the following embodiments are mere examples, and thus are not intended to limit the present disclosure. Of the structural components described in the following embodiments, structural components not recited in any one of the independent claims that indicate the broadest concepts will be described as optional structural components.

[0039]

EMBODIMENT 1

First, the data structure of encoded three-dimensional data (hereinafter also referred to as encoded data) according to the present embodiment will be described. FIG. 1 is a diagram showing the structure of encoded three-dimensional data according to the present

embodiment.

[0040]

In the present embodiment, a three-dimensional space is divided into spaces (SPCs), which correspond to pictures in moving picture encoding, and the three-dimensional data is encoded on a SPC-by-SPC basis. Each SPC is further divided into volumes (VLMs), which correspond to macroblocks, etc. in moving picture encoding, and predictions and transforms are performed on a VLM-by-VLM basis. Each volume includes a plurality of voxels (VXLs), each being a minimum unit in which position coordinates are associated. Note that prediction is a process of generating predictive three-dimensional data analogous to a current processing unit by referring to another processing unit, and encoding a differential between the predictive three-dimensional data and the current processing unit, as in the case of predictions performed on two-dimensional images. Such prediction includes not only spatial prediction in which another prediction unit corresponding to the same time is referred to, but also temporal prediction in which a prediction unit corresponding to a different time is referred to.

[0041]

When encoding a three-dimensional space represented by point group data such as a point cloud, for example, the three-dimensional data encoding device (hereinafter also referred to as the encoding device) encodes the points in the point group or points included in the respective voxels in a collective manner, in accordance with a voxel size. Finer voxels enable a highly-precise representation of the three-dimensional shape of a point group, while larger voxels enable a rough representation of the three-dimensional shape of a point group.

[0042]

Note that the following describes the case where three-dimensional data is a point cloud, but three-dimensional data is not limited to a point cloud, and thus three-dimensional data of any format may be employed.

[0043]

Also note that voxels with a hierarchical structure may be used. In such a case, when the hierarchy includes n levels, whether a sampling point is included in the n-1th level or its lower levels (the lower levels of the n-th level) may be sequentially indicated. For example, when only the n-th level is decoded, and the n-1th level or its lower levels include a sampling point, the n-th level can be decoded on the assumption that a sampling point is included at the center of a voxel in the n-th level.

[0044]

Also, the encoding device obtains point group data, using, for example, a distance sensor, a stereo camera, a monocular camera, a gyroscope sensor, or an inertial sensor.

[0045]

As in the case of moving picture encoding, each SPC is classified into one of at least the three prediction structures that include: intra SPC (I-SPC), which is individually decodable; predictive SPC (P-SPC) capable of only a unidirectional reference; and bidirectional SPC (B-SPC) capable of bidirectional references. Each SPC includes two types of time information: decoding time and display time.

[0046]

Furthermore, as shown in FIG. 1, a processing unit that includes a plurality of SPCs is a group of spaces (GOS), which is a random access unit. Also, a processing unit that includes a plurality of GOSs is a world (WLD).

[0047]

The spatial region occupied by each world is associated with an absolute position on earth, by use of, for example, GPS, or latitude and longitude information. Such position information is stored as meta-information. Note that meta-information may be included in encoded data, or may be transmitted separately from the encoded data.

[0048]

Also, inside a GOS, all SPCs may be three-dimensionally adjacent to one another, or there may be a SPC that is not three-dimensionally adjacent to another SPC.

[0049]

Note that the following also describes processes such as encoding, decoding, and reference to be performed on three-dimensional data included in processing units such as GOS, SPC, and VLM, simply as performing encoding/to encode, decoding/to decode, referring to, etc. on a processing unit. Also note that three-dimensional data included in a processing unit includes, for example, at least one pair of a spatial position such as three-dimensional coordinates and an attribute value such as color information.

[0050]

Next, the prediction structures among SPCs in a GOS will be described. A plurality of SPCs in the same GOS or a plurality of VLMs in the same SPC occupy mutually different spaces, while having the same time information (the decoding time and the display time).

[0051]

A SPC in a GOS that comes first in the decoding order is an I-SPC. GOSs come in two types: closed GOS and open GOS. A closed GOS is a GOS in which all SPCs in the GOS are decodable when decoding starts from the first I-SPC. Meanwhile, an open GOS is a GOS in which a different GOS is referred to in one or more SPCs preceding the first I-SPC in the GOS in the display time, and thus cannot be singly decoded.

[0052]

Note that in the case of encoded data of map information, for example, a WLD is sometimes decoded in the backward direction, which is opposite to the encoding order, and thus backward reproduction is difficult when GOSs are interdependent. In such a case, a closed GOS is basically used.

[0053]

Each GOS has a layer structure in height direction, and SPCs are sequentially encoded or decoded from SPCs in the bottom layer.

[0054]

FIG. 2 is a diagram showing an example of prediction structures among SPCs that belong to the lowermost layer in a GOS. FIG. 3 is a diagram showing an example of prediction structures among layers.

[0055]

A GOS includes at least one I-SPC. Of the objects in a three-dimensional space, such as a person, an animal, a car, a bicycle, a signal, and a building serving as a landmark, a small-sized object is especially effective when encoded as an I-SPC. When decoding a GOS at a low throughput or at a high speed, for example, the three-dimensional data decoding device (hereinafter also referred to as the decoding device) decodes only I-SPC(s) in the GOS.

[0056]

The encoding device may also change the encoding interval or the appearance frequency of I-SPCs, depending on the degree of sparseness and denseness of the objects in a WLD.

[0057]

In the structure shown in FIG. 3, the encoding device or the decoding device encodes or decodes a plurality of layers sequentially from the bottom layer (layer 1). This increases the priority of data on the ground and its vicinity, which involve a larger amount of information, when, for example, a self-driving car is concerned.

[0058]

Regarding encoded data used for a drone, for example, encoding or decoding may be performed sequentially from SPCs in the top layer in a GOS in height direction.

[0059]

The encoding device or the decoding device may also encode or decode a plurality of layers in a manner that the decoding device can have a rough grasp of a GOS first, and then the resolution is gradually increased. The encoding device or the decoding device may perform encoding or decoding in the order of layers 3, 8, 1, 9..., for example.

[0060]

Next, the handling of static objects and dynamic objects will be described.

[0061]

A three-dimensional space includes scenes or still objects such as a building and a road (hereinafter collectively referred to as static objects), and objects with motion such as a car and a person (hereinafter collectively referred to as dynamic objects). Object detection is separately performed by, for example, extracting keypoints from point cloud data, or from video of a camera such as a stereo camera. In this description, an example method of encoding a dynamic object will be described.

[0062]

A first method is a method in which a static object and a dynamic object are encoded without distinction. A second method is a method in which a distinction is made between a static object and a dynamic object on the basis of identification information.

[0063]

For example, a GOS is used as an identification unit. In such a case, a distinction is made between a GOS that includes SPCs constituting a static object and a GOS that includes SPCs constituting a dynamic object, on the basis of identification information stored in the encoded data or stored separately from the encoded data.

[0064]

Alternatively, a SPC may be used as an identification unit. In such a case, a distinction is made between a SPC that includes VLMs constituting a static object and a SPC that includes VLMs constituting a dynamic object, on the basis of the identification information thus described.

[0065]

Alternatively, a VLM or a VXL may be used as an identification unit. In such a case, a distinction is made between a VLM or a VXL that includes a static object and a VLM or a VXL that includes a dynamic object, on the basis of the identification information thus

described.

[0066]

The encoding device may also encode a dynamic object as at least one VLM or SPC, and may encode a VLM or a SPC including a static object and a SPC including a dynamic object as mutually different GOSs. When the GOS size is variable depending on the size of a dynamic object, the encoding device separately stores the GOS size as meta-information.

[0067]

The encoding device may also encode a static object and a dynamic object separately from each other, and may superimpose the dynamic object onto a world constituted by static objects. In such a case, the dynamic object is constituted by at least one SPC, and each SPC is associated with at least one SPC constituting the static object onto which the each SPC is to be superimposed. Note that a dynamic object may be represented not by SPC(s) but by at least one VLM or VXL.

[0068]

The encoding device may also encode a static object and a dynamic object as mutually different streams.

[0069]

The encoding device may also generate a GOS that includes at least one SPC constituting a dynamic object. The encoding device may further set the size of a GOS including a dynamic object (GOS_M) and the size of a GOS including a static object corresponding to the spatial region of GOS_M at the same size (such that the same spatial region is occupied). This enables superimposition to be performed on a GOS-by-GOS basis.

[0070]

SPC(s) included in another encoded GOS may be referred to in a P-SPC or a B-SPC constituting a dynamic object. In the case where the position of a dynamic object temporally changes, and the same dynamic object is encoded as an object in a GOS

corresponding to a different time, referring to SPC(s) across GOSs is effective in terms of compression rate.

[0071]

The first method and the second method may be selected in accordance with the intended use of encoded data. When encoded three-dimensional data is used as a map, for example, a dynamic object is desired to be separated, and thus the encoding device uses the second method. Meanwhile, the encoding device uses the first method when the separation of a dynamic object is not required such as in the case where three-dimensional data of an event such as a concert and a sports event is encoded.

[0072]

The decoding time and the display time of a GOS or a SPC are storable in encoded data or as meta-information. All static objects may have the same time information. In such a case, the decoding device may determine the actual decoding time and display time. Alternatively, a different value may be assigned to each GOS or SPC as the decoding time, and the same value may be assigned as the display time. Furthermore, as in the case of the decoder model in moving picture encoding such as Hypothetical Reference Decoder (HRD) compliant with HEVC, a model may be employed that ensures that a decoder can perform decoding without fail by having a buffer of a predetermined size and by reading a bitstream at a predetermined bit rate in accordance with the decoding times.

[0073]

Next, the topology of GOSs in a world will be described. The coordinates of the three-dimensional space in a world are represented by the three coordinate axes (x axis, y axis, and z axis) that are orthogonal to one another. A predetermined rule set for the encoding order of GOSs enables encoding to be performed such that spatially adjacent GOSs are contiguous in the encoded data. In an example shown in FIG. 4, for example, GOSs in the x and z planes are successively encoded. After the completion of encoding all GOSs in certain x and z planes, the value of the y axis is updated. Stated differently, the world

expands in the y axis direction as the encoding progresses. The GOS index numbers are set in accordance with the encoding order.

[0074]

Here, the three-dimensional spaces in the respective worlds are previously associated one-to-one with absolute geographical coordinates such as GPS coordinates or latitude/longitude coordinates. Alternatively, each three-dimensional space may be represented as a position relative to a previously set reference position. The directions of the x axis, the y axis, and the z axis in the three-dimensional space are represented by directional vectors that are determined on the basis of the latitudes and the longitudes, etc. Such directional vectors are stored together with the encoded data as meta-information.

[0075]

GOSs have a fixed size, and the encoding device stores such size as meta-information. The GOS size may be changed depending on, for example, whether it is an urban area or not, or whether it is inside or outside of a room. Stated differently, the GOS size may be changed in accordance with the amount or the attributes of objects with information values. Alternatively, in the same world, the encoding device may adaptively change the GOS size or the interval between I-SPCs in GOSs in accordance with the object density, etc. For example, the encoding device sets the GOS size to smaller and the interval between I-SPCs in GOSs to shorter, as the object density is higher.

[0076]

In an example shown in FIG. 5, to enable random access with a finer granularity, a GOS with a high object density is partitioned into the regions of the third to tenth GOSs. Note that the seventh to tenth GOSs are located behind the third to sixth GOSs.

[0077]

Next, the structure and the operation flow of the three-dimensional data encoding device according to the present embodiment will be described. FIG. 6 is a block diagram of three-dimensional data encoding device 100 according to the present embodiment. FIG. 7 is

a flowchart of an example operation performed by three-dimensional data encoding device 100.

[0078]

Three-dimensional data encoding device 100 shown in FIG. 6 encodes three-dimensional data 111, thereby generating encoded three-dimensional data 112. Such three-dimensional data encoding device 100 includes obtainer 101, encoding region determiner 102, divider 103, and encoder 104.

[0079]

As shown in FIG. 7, first, obtainer 101 obtains three-dimensional data 111, which is point group data (S101).

[0080]

Next, encoding region determiner 102 determines a current region for encoding from among spatial regions corresponding to the obtained point group data (S102). For example, in accordance with the position of a user or a vehicle, encoding region determiner 102 determines, as the current region, a spatial region around such position.

[0081]

Next, divider 103 divides the point group data included in the current region into processing units. The processing units here means units such as GOSs and SPCs described above. The current region here corresponds to, for example, a world described above. More specifically, divider 103 divides the point group data into processing units on the basis of a predetermined GOS size, or the presence/absence/size of a dynamic object (S103). Divider 103 further determines the starting position of the SPC that comes first in the encoding order in each GOS.

[0082]

Next, encoder 104 sequentially encodes a plurality of SPCs in each GOS, thereby generating encoded three-dimensional data 112 (S104).

[0083]

Note that although an example is described here in which the current region is divided into GOSs and SPCs, after which each GOS is encoded, the processing steps are not limited to this order. For example, steps may be employed in which the structure of a single GOS is determined, which is followed by the encoding of such GOS, and then the structure of the subsequent GOS is determined.

[0084]

As thus described, three-dimensional data encoding device 100 encodes three-dimensional data 111, thereby generating encoded three-dimensional data 112. More specifically, three-dimensional data encoding device 100 divides three-dimensional data into first processing units (GOSs), each being a random access unit and being associated with three-dimensional coordinates, divides each of the first processing units (GOSs) into second processing units (SPCs), and divides each of the second processing units (SPCs) into third processing units (VLMs). Each of the third processing units (VLMs) includes at least one voxel (VXL), which is the minimum unit in which position information is associated.

[0085]

Next, three-dimensional data encoding device 100 encodes each of the first processing units (GOSs), thereby generating encoded three-dimensional data 112. More specifically, three-dimensional data encoding device 100 encodes each of the second processing units (SPCs) in each of the first processing units (GOSs). Three-dimensional data encoding device 100 further encodes each of the third processing units (VLMs) in each of the second processing units (SPCs).

[0086]

When a current first processing unit (GOS) is a closed GOS, for example, three-dimensional data encoding device 100 encodes a current second processing unit (SPC) included in such current first processing unit (GOS) by referring to another second processing unit (SPC) included in the current first processing unit (GOS). Stated differently, three-dimensional data encoding device 100 refers to no second processing unit (SPC)

included in a first processing unit (GOS) that is different from the current first processing unit (GOS).

[0087]

Meanwhile, when a current first processing unit (GOS) is an open GOS, three-dimensional data encoding device 100 encodes a current second processing unit (SPC) included in such current first processing unit (GOS) by referring to another second processing unit (SPC) included in the current first processing unit (GOS) or a second processing unit (SPC) included in a first processing unit (GOS) that is different from the current first processing unit (GOS).

[0088]

Also, three-dimensional data encoding device 100 selects, as the type of a current second processing unit (SPC), one of the following: a first type (I-SPC) in which another second processing unit (SPC) is not referred to; a second type (P-SPC) in which another single second processing unit (SPC) is referred to; and a third type in which other two second processing units (SPC) are referred to. Three-dimensional data encoding device 100 encodes the current second processing unit (SPC) in accordance with the selected type.

[0089]

Next, the structure and the operation flow of the three-dimensional data decoding device according to the present embodiment will be described. FIG. 8 is a block diagram of three-dimensional data decoding device 200 according to the present embodiment. FIG. 9 is a flowchart of an example operation performed by three-dimensional data decoding device 200.

[0090]

Three-dimensional data decoding device 200 shown in FIG. 8 decodes encoded three-dimensional data 211, thereby generating decoded three-dimensional data 212. Encoded three-dimensional data 211 here is, for example, encoded three-dimensional data 112 generated by three-dimensional data encoding device 100. Such three-dimensional data

decoding device 200 includes obtainer 201, decoding start GOS determiner 202, decoding SPC determiner 203, and decoder 204.

[0091]

First, obtainer 201 obtains encoded three-dimensional data 211 (S201). Next, decoding start GOS determiner 202 determines a current GOS for decoding (S202). More specifically, decoding start GOS determiner 202 refers to meta-information stored in encoded three-dimensional data 211 or stored separately from the encoded three-dimensional data to determine, as the current GOS, a GOS that includes a SPC corresponding to the spatial position, the object, or the time from which decoding is to start.

[0092]

Next, decoding SPC determiner 203 determines the type(s) (I, P, and/or B) of SPCs to be decoded in the GOS (S203). For example, decoding SPC determiner 203 determines whether to (1) decode only I-SPC(s), (2) to decode I-SPC(s) and P-SPCs, or (3) to decode SPCs of all types. Note that the present step may not be performed, when the type(s) of SPCs to be decoded are previously determined such as when all SPCs are previously determined to be decoded.

[0093]

Next, decoder 204 obtains an address location within encoded three-dimensional data 211 from which a SPC that comes first in the GOS in the decoding order (the same as the encoding order) starts. Decoder 204 obtains the encoded data of the first SPC from the address location, and sequentially decodes the SPCs from such first SPC (S204). Note that the address location is stored in the meta-information, etc.

[0094]

Three-dimensional data decoding device 200 decodes decoded three-dimensional data 212 as thus described. More specifically, three-dimensional data decoding device 200 decodes each encoded three-dimensional data 211 of the first processing units (GOSs), each being a random access unit and being associated with three-dimensional coordinates, thereby

generating decoded three-dimensional data 212 of the first processing units (GOSs). Even more specifically, three-dimensional data decoding device 200 decodes each of the second processing units (SPCs) in each of the first processing units (GOSs). Three-dimensional data decoding device 200 further decodes each of the third processing units (VLMs) in each of the second processing units (SPCs).

[0095]

The following describes meta-information for random access. Such meta-information is generated by three-dimensional data encoding device 100, and included in encoded three-dimensional data 112 (211).

[0096]

In the conventional random access for a two-dimensional moving picture, decoding starts from the first frame in a random access unit that is close to a specified time. Meanwhile, in addition to times, random access to spaces (coordinates, objects, etc.) is assumed to be performed in a world.

[0097]

To enable random access to at least three elements of coordinates, objects, and times, tables are prepared that associate the respective elements with the GOS index numbers. Furthermore, the GOS index numbers are associated with the addresses of the respective first I-SPCs in the GOSs. FIG. 10 is a diagram showing example tables included in the meta-information. Note that not all the tables shown in FIG. 10 are required to be used, and thus at least one of the tables is used.

[0098]

The following describes an example in which random access is performed from coordinates as a starting point. To access the coordinates (x2, y2, and z2), the coordinates-GOS table is first referred to, which indicates that the point corresponding to the coordinates (x2, y2, and z2) is included in the second GOS. Next, the GOS-address table is referred to, which indicates that the address of the first I-SPC in the second GOS is addr(2).

As such, decoder 204 obtains data from this address to start decoding.

[0099]

Note that the addresses may either be logical addresses or physical addresses of an HDD or a memory. Alternatively, information that identifies file segments may be used instead of addresses. File segments are, for example, units obtained by segmenting at least one GOS, etc.

[0100]

When an object spans across a plurality of GOSs, the object-GOS table may show a plurality of GOSs to which such object belongs. When such plurality of GOSs are closed GOSs, the encoding device and the decoding device can perform encoding or decoding in parallel. Meanwhile, when such plurality of GOSs are open GOSs, a higher compression efficiency is achieved by the plurality of GOSs referring to each other.

[0101]

Example objects include a person, an animal, a car, a bicycle, a signal, and a building serving as a landmark. For example, three-dimensional data encoding device 100 extracts keypoints specific to an object from a three-dimensional point cloud, etc., when encoding a world, and detects the object on the basis of such keypoints to set the detected object as a random access point.

[0102]

As thus described, three-dimensional data encoding device 100 generates first information indicating a plurality of first processing units (GOSs) and the three-dimensional coordinates associated with the respective first processing units (GOSs). Encoded three-dimensional data 112 (211) includes such first information. The first information further indicates at least one of objects, times, and data storage locations that are associated with the respective first processing units (GOSs).

[0103]

Three-dimensional data decoding device 200 obtains the first information from

encoded three-dimensional data 211. Using such first information, three-dimensional data decoding device 200 identifies encoded three-dimensional data 211 of the first processing unit that corresponds to the specified three-dimensional coordinates, object, or time, and decodes encoded three-dimensional data 211.

[0104]

The following describes an example of other meta-information. In addition to the meta-information for random access, three-dimensional data encoding device 100 may also generate and store meta-information as described below, and three-dimensional data decoding device 200 may use such meta-information at the time of decoding.

[0105]

When three-dimensional data is used as map information, for example, a profile is defined in accordance with the intended use, and information indicating such profile may be included in meta-information. For example, a profile is defined for an urban or a suburban area, or for a flying object, and the maximum or minimum size, etc. of a world, a SPC or a VLM, etc. is defined in each profile. For example, more detailed information is required for an urban area than for a suburban area, and thus the minimum VLM size is set to small.

[0106]

The meta-information may include tag values indicating object types. Each of such tag values is associated with VLMs, SPCs, or GOSs that constitute an object. For example, a tag value may be set for each object type in a manner, for example, that the tag value "0" indicates "person," the tag value "1" indicates "car," and the tag value "2" indicates "signal." Alternatively, when an object type is hard to judge, or such judgment is not required, a tag value may be used that indicates the size or the attribute indicating, for example, whether an object is a dynamic object or a static object.

[0107]

The meta-information may also include information indicating a range of the spatial region occupied by a world.

[0108]

The meta-information may also store the SPC or VXL size as header information common to the whole stream of the encoded data or to a plurality of SPCs, such as SPCs in a GOS.

[0109]

The meta-information may also include identification information on a distance sensor or a camera that has been used to generate a point cloud, or information indicating the positional accuracy of a point group in the point cloud.

[0110]

The meta-information may also include information indicating whether a world is made only of static objects or includes a dynamic object.

[0111]

The following describes variations of the present embodiment.

[0112]

The encoding device or the decoding device may encode or decode two or more mutually different SPCs or GOSs in parallel. GOSs to be encoded or decoded in parallel can be determined on the basis of meta-information, etc. indicating the spatial positions of the GOSs.

[0113]

When three-dimensional data is used as a spatial map for use by a car or a flying object, etc. in traveling, or for creation of such a spatial map, for example, the encoding device or the decoding device may encode or decode GOSs or SPCs included in a space that is identified on the basis of GPS information, the route information, the zoom magnification, etc.

[0114]

The decoding device may also start decoding sequentially from a space that is close to the self-location or the traveling route. The encoding device or the decoding device may

give a lower priority to a space distant from the self-location or the traveling route than the priority of a nearby space to encode or decode such distant place. To "give a lower priority" means here, for example, to lower the priority in the processing sequence, to decrease the resolution (to apply decimation in the processing), or to lower the image quality (to increase the encoding efficiency by, for example, setting the quantization step to larger).

[0115]

When decoding encoded data that is hierarchically encoded in a space, the decoding device may decode only the bottom level in the hierarchy.

[0116]

The decoding device may also start decoding preferentially from the bottom level of the hierarchy in accordance with the zoom magnification or the intended use of the map.

[0117]

For self-location estimation or object recognition, etc. involved in the self-driving of a car or a robot, the encoding device or the decoding device may encode or decode regions at a lower resolution, except for a region that is lower than or at a specified height from the ground (the region to be recognized).

[0118]

The encoding device may also encode point clouds representing the spatial shapes of a room interior and a room exterior separately. For example, the separation of a GOS representing a room interior (interior GOS) and a GOS representing a room exterior (exterior GOS) enables the decoding device to select a GOS to be decoded in accordance with a viewpoint location, when using the encoded data.

[0119]

The encoding device may also encode an interior GOS and an exterior GOS having close coordinates so that such GOSs come adjacent to each other in an encoded stream. For example, the encoding device associates the identifiers of such GOSs with each other, and stores information indicating the associated identifiers into the meta-information that is stored

in the encoded stream or stored separately. This enables the decoding device to refer to the information in the meta-information to identify an interior GOS and an exterior GOS having close coordinates.

[0120]

The encoding device may also change the GOS size or the SPC size depending on whether a GOS is an interior GOS or an exterior GOS. For example, the encoding device sets the size of an interior GOS to smaller than the size of an exterior GOS. The encoding device may also change the accuracy of extracting keypoints from a point cloud, or the accuracy of detecting objects, for example, depending on whether a GOS is an interior GOS or an exterior GOS.

[0121]

The encoding device may also add, to encoded data, information by which the decoding device displays objects with a distinction between a dynamic object and a static object. This enables the decoding device to display a dynamic object together with, for example, a red box or letters for explanation. Note that the decoding device may display only a red box or letters for explanation, instead of a dynamic object. The decoding device may also display more particular object types. For example, a red box may be used for a car, and a yellow box may be used for a person.

[0122]

The encoding device or the decoding device may also determine whether to encode or decode a dynamic object and a static object as a different SPC or GOS, in accordance with, for example, the appearance frequency of dynamic objects or a ratio between static objects and dynamic objects. For example, when the appearance frequency or the ratio of dynamic objects exceeds a threshold, a SPC or a GOS including a mixture of a dynamic object and a static object is accepted, while when the appearance frequency or the ratio of dynamic objects is below a threshold, a SPC or GOS including a mixture of a dynamic object and a static object is unaccepted.

[0123]

When detecting a dynamic object not from a point cloud but from two-dimensional image information of a camera, the encoding device may separately obtain information for identifying a detection result (box or letters) and the object position, and encode these items of information as part of the encoded three-dimensional data. In such a case, the decoding device superimposes auxiliary information (box or letters) indicating the dynamic object onto a resultant of decoding a static object to display it.

[0124]

The encoding device may also change the sparseness and denseness of VXLs or VLMs in a SPC in accordance with the degree of complexity of the shape of a static object. For example, the encoding device sets VXLs or VLMs at a higher density as the shape of a static object is more complex. The encoding device may further determine a quantization step, etc. for quantizing spatial positions or color information in accordance with the sparseness and denseness of VXLs or VLMs. For example, the encoding device sets the quantization step to smaller as the density of VXLs or VLMs is higher.

[0125]

As described above, the encoding device or the decoding device according to the present embodiment encodes or decodes a space on a SPC-by-SPC basis that includes coordinate information.

[0126]

Furthermore, the encoding device and the decoding device perform encoding or decoding on a volume-by-volume basis in a SPC. Each volume includes a voxel, which is the minimum unit in which position information is associated.

[0127]

Also, using a table that associates the respective elements of spatial information including coordinates, objects, and times with GOSs or using a table that associates these elements with each other, the encoding device and the decoding device associate any ones of

the elements with each other to perform encoding or decoding. The decoding device uses the values of the selected elements to determine the coordinates, and identifies a volume, a voxel, or a SPC from such coordinates to decode a SPC including such volume or voxel, or the identified SPC.

[0128]

Furthermore, the encoding device determines a volume, a voxel, or a SPC that is selectable in accordance with the elements, through extraction of keypoints and object recognition, and encodes the determined volume, voxel, or SPC, as a volume, a voxel, or a SPC to which random access is possible.

[0129]

SPCs are classified into three types: I-SPC that is singly encodable or decodable; P-SPC that is encoded or decoded by referring to any one of the processed SPCs; and B-SPC that is encoded or decoded by referring to any two of the processed SPCs.

[0130]

At least one volume corresponds to a static object or a dynamic object. A SPC including a static object and a SPC including a dynamic object are encoded or decoded as mutually different GOSs. Stated differently, a SPC including a static object and a SPC including a dynamic object are assigned to different GOSs.

[0131]

Dynamic objects are encoded or decoded on an object-by-object basis, and are associated with at least one SPC including a static object. Stated differently, a plurality of dynamic objects are individually encoded, and the obtained encoded data of the dynamic objects is associated with a SPC including a static object.

[0132]

The encoding device and the decoding device give an increased priority to I-SPC(s) in a GOS to perform encoding or decoding. For example, the encoding device performs encoding in a manner that prevents the degradation of I-SPCs (in a manner that enables the

original three-dimensional data to be reproduced with a higher fidelity after decoded). The decoding device decodes, for example, only I-SPCs.

[0133]

The encoding device may change the frequency of using I-SPCs depending on the sparseness and denseness or the number (amount) of the objects in a world to perform encoding. Stated differently, the encoding device changes the frequency of selecting I-SPCs depending on the number or the sparseness and denseness of the objects included in the three-dimensional data. For example, the encoding device uses I-SPCs at a higher frequency as the density of the objects in a world is higher.

[0134]

The encoding device also sets random access points on a GOS-by-GOS basis, and stores information indicating the spatial regions corresponding to the GOSs into the header information.

[0135]

The encoding device uses, for example, a default value as the spatial size of a GOS. Note that the encoding device may change the GOS size depending on the number (amount) or the sparseness and denseness of objects or dynamic objects. For example, the encoding device sets the spatial size of a GOS to smaller as the density of objects or dynamic objects is higher or the number of objects or dynamic objects is greater.

[0136]

Also, each SPC or volume includes a keypoint group that is derived by use of information obtained by a sensor such as a depth sensor, a gyroscope sensor, or a camera sensor. The coordinates of the keypoints are set at the central positions of the respective voxels. Furthermore, finer voxels enable highly accurate position information.

[0137]

The keypoint group is derived by use of a plurality of pictures. A plurality of pictures include at least two types of time information: the actual time information and the

same time information common to a plurality of pictures that are associated with SPCs (for example, the encoding time used for rate control, etc.).

[0138]

Also, encoding or decoding is performed on a GOS-by-GOS basis that includes at least one SPC.

[0139]

The encoding device and the decoding device predict P-SPCs or B-SPCs in a current GOS by referring to SPCs in a processed GOS.

[0140]

Alternatively, the encoding device and the decoding device predict P-SPCs or B-SPCs in a current GOS, using the processed SPCs in the current GOS, without referring to a different GOS.

[0141]

Furthermore, the encoding device and the decoding device transmit or receive an encoded stream on a world-by-world basis that includes at least one GOS.

[0142]

Also, a GOS has a layer structure in one direction at least in a world, and the encoding device and the decoding device start encoding or decoding from the bottom layer. For example, a random accessible GOS belongs to the lowermost layer. A GOS that belongs to the same layer or a lower layer is referred to in a GOS that belongs to an upper layer. Stated differently, a GOS is spatially divided in a predetermined direction in advance to have a plurality of layers, each including at least one SPC. The encoding device and the decoding device encode or decode each SPC by referring to a SPC included in the same layer as the each SPC or a SPC included in a layer lower than that of the each SPC.

[0143]

Also, the encoding device and the decoding device successively encode or decode GOSs on a world-by-world basis that includes such GOSs. In so doing, the encoding device

and the decoding device write or read out information indicating the order (direction) of encoding or decoding as metadata. Stated differently, the encoded data includes information indicating the order of encoding a plurality of GOSs.

[0144]

The encoding device and the decoding device also encode or decode mutually different two or more SPCs or GOSs in parallel.

[0145]

Furthermore, the encoding device and the decoding device encode or decode the spatial information (coordinates, size, etc.) on a SPC or a GOS.

[0146]

The encoding device and the decoding device encode or decode SPCs or GOSs included in an identified space that is identified on the basis of external information on the self-location or/and region size, such as GPS information, route information, or magnification.

[0147]

The encoding device or the decoding device gives a lower priority to a space distant from the self-location than the priority of a nearby space to perform encoding or decoding.

[0148]

The encoding device sets a direction at one of the directions in a world, in accordance with the magnification or the intended use, to encode a GOS having a layer structure in such direction. Also, the decoding device decodes a GOS having a layer structure in one of the directions in a world that has been set in accordance with the magnification or the intended use, preferentially from the bottom layer.

[0149]

The encoding device changes the accuracy of extracting keypoints, the accuracy of recognizing objects, or the size of spatial regions, etc. included in a SPC, depending on whether an object is an interior object or an exterior object. Note that the encoding device

and the decoding device encode or decode an interior GOS and an exterior GOS having close coordinates in a manner that these GOSs come adjacent to each other in a world, and associates their identifiers with each other for encoding and decoding.

[0150]

EMBODIMENT 2

When using encoded data of a point cloud in an actual device or service, it is desirable that necessary information be transmitted/received in accordance with the intended use to reduce the network bandwidth. However, there has been no such functionality in the structure of encoding three-dimensional data, nor an encoding method therefor.

[0151]

The present embodiment describes a three-dimensional data encoding method and a three-dimensional data encoding device for providing the functionality of transmitting/receiving only necessary information in encoded data of a three-dimensional point cloud in accordance with the intended use, as well as a three-dimensional data decoding method and a three-dimensional data decoding device for decoding such encoded data.

[0152]

A voxel (VXL) with a feature greater than or equal to a given amount is defined as a feature voxel (FVXL), and a world (WLD) constituted by FVXLs is defined as a sparse world (SWLD). FIG. 11 is a diagram showing example structures of a sparse world and a world. A SWLD includes: FGOSs, each being a GOS constituted by FVXLs; FSPCs, each being a SPC constituted by FVXLs; and FVLMs, each being a VLM constituted by FVXLs. The data structure and prediction structure of a FGOS, a FSPC, and a FVLM may be the same as those of a GOS, a SPC, and a VLM.

[0153]

A feature represents the three-dimensional position information on a VXL or the visible-light information on the position of a VXL. A large number of features are detected especially at a corner, an edge, etc. of a three-dimensional object. More specifically, such a

feature is a three-dimensional feature or a visible-light feature as described below, but may be any feature that represents the position, luminance, or color information, etc. on a VXL.

[0154]

Used as three-dimensional features are signature of histograms of orientations (SHOT) features, point feature histograms (PFH) features, or point pair feature (PPF) features.

[0155]

SHOT features are obtained by dividing the periphery of a VXL, and calculating an inner product of the reference point and the normal vector of each divided region to represent the calculation result as a histogram. SHOT features are characterized by a large number of dimensions and high-level feature representation.

[0156]

PFH features are obtained by selecting a large number of two point pairs in the vicinity of a VXL, and calculating the normal vector, etc. from each two point pair to represent the calculation result as a histogram. PFH features are histogram features, and thus are characterized by robustness against a certain extent of disturbance and also high-level feature representation.

[0157]

PPF features are obtained by using a normal vector, etc. for each two points of VXLs. PPF features, for which all VXLs are used, has robustness against occlusion.

[0158]

Used as visible-light features are scale-invariant feature transform (SIFT), speeded up robust features (SURF), or histogram of oriented gradients (HOG), etc. that use information on an image such as luminance gradient information.

[0159]

A SWLD is generated by calculating the above-described features of the respective VXLs in a WLD to extract FVXLs. Here, the SWLD may be updated every time the WLD

is updated, or may be regularly updated after the elapse of a certain period of time, regardless of the timing at which the WLD is updated.

[0160]

A SWLD may be generated for each type of features. For example, different SWLDs may be generated for the respective types of features, such as SWLD1 based on SHOT features and SWLD2 based on SIFT features so that SWLDs are selectively used in accordance with the intended use. Also, the calculated feature of each FVXL may be held in each FVXL as feature information.

[0161]

Next, the usage of a sparse world (SWLD) will be described. A SWLD includes only feature voxels (FVXLs), and thus its data size is smaller in general than that of a WLD that includes all VXLs.

[0162]

In an application that utilizes features for a certain purpose, the use of information on a SWLD instead of a WLD reduces the time required to read data from a hard disk, as well as the bandwidth and the time required for data transfer over a network. For example, a WLD and a SWLD are held in a server as map information so that map information to be sent is selected between the WLD and the SWLD in accordance with a request from a client. This reduces the network bandwidth and the time required for data transfer. More specific examples will be described below.

[0163]

FIG. 12 and FIG. 13 are diagrams showing usage examples of a SWLD and a WLD. As FIG. 12 shows, when client 1, which is a vehicle-mounted device, requires map information to use it for self-location determination, client 1 sends to a server a request for obtaining map data for self-location estimation (S301). The server sends to client 1 the SWLD in response to the obtainment request (S302). Client 1 uses the received SWLD to determine the self-location (S303). In so doing, client 1 obtains VXL information on the

periphery of client 1 through various means including a distance sensor such as a rangefinder, as well as a stereo camera and a combination of a plurality of monocular cameras. Client 1 then estimates the self-location information from the obtained VXL information and the SWLD. Here, the self-location information includes three-dimensional position information, orientation, etc. of client 1.

[0164]

As FIG. 13 shows, when client 2, which is a vehicle-mounted device, requires map information to use it for rendering a map such as a three-dimensional map, client 2 sends to the server a request for obtaining map data for map rendering (S311). The server sends to client 2 the WLD in response to the obtainment request (S312). Client 2 uses the received WLD to render a map (S313). In so doing, client 2 uses, for example, an image client 2 has captured by a visible-light camera, etc. and the WLD obtained from the server to create a rendering image, and renders such created image onto a screen of a car navigation system, etc.

[0165]

As described above, the server sends to a client a SWLD when the features of the respective VXLs are mainly required such as in the case of self-location estimation, and sends to a client a WLD when detailed VXL information is required such as in the case of map rendering. This allows for an efficient sending/receiving of map data.

[0166]

Note that a client may self-judge which one of a SWLD and a WLD is necessary, and request the server to send a SWLD or a WLD. Also, the server may judge which one of a SWLD and a WLD to send in accordance with the status of the client or a network.

[0167]

Next, a method will be described of switching the sending/receiving between a sparse world (SWLD) and a world (WLD).

[0168]

Whether to receive a WLD or a SWLD may be switched in accordance with the network bandwidth. FIG. 14 is a diagram showing an example operation in such case. For example, when a low-speed network is used that limits the usable network bandwidth, such as in a Long-Term Evolution (LTE) environment, a client accesses the server over a low-speed network (S321), and obtains the SWLD from the server as map information (S322). Meanwhile, when a high-speed network is used that has an adequately broad network bandwidth, such as in a WiFi environment, a client accesses the server over a high-speed network (S323), and obtains the WLD from the server (S324). This enables the client to obtain appropriate map information in accordance with the network bandwidth such client is using.

[0169]

More specifically, a client receives the SWLD over an LTE network when in outdoors, and obtains the WLD over a WiFi network when in indoors such as in a facility. This enables the client to obtain more detailed map information on indoor environment.

[0170]

As described above, a client may request for a WLD or a SWLD in accordance with the bandwidth of a network such client is using. Alternatively, the client may send to the server information indicating the bandwidth of a network such client is using, and the server may send to the client data (the WLD or the SWLD) suitable for such client in accordance with the information. Alternatively, the server may identify the network bandwidth the client is using, and send to the client data (the WLD or the SWLD) suitable for such client.

[0171]

Also, whether to receive a WLD or a SWLD may be switched in accordance with the speed of traveling. FIG. 15 is a diagram showing an example operation in such case. For example, when traveling at a high speed (S331), a client receives the SWLD from the server (S332). Meanwhile, when traveling at a low speed (S333), the client receives the WLD from the server (S334). This enables the client to obtain map information suitable to

the speed, while reducing the network bandwidth. More specifically, when traveling on an expressway, the client receives the SWLD with a small data amount, which enables the update of rough map information at an appropriate speed. Meanwhile, when traveling on a general road, the client receives the WLD, which enables the obtainment of more detailed map information.

[0172]

As described above, the client may request the server for a WLD or a SWLD in accordance with the traveling speed of such client. Alternatively, the client may send to the server information indicating the traveling speed of such client, and the server may send to the client data (the WLD or the SWLD) suitable to such client in accordance with the information. Alternatively, the server may identify the traveling speed of the client to send data (the WLD or the SWLD) suitable to such client.

[0173]

Also, the client may obtain, from the server, a SWLD first, from which the client may obtain a WLD of an important region. For example, when obtaining map information, the client first obtains a SWLD for rough map information, from which the client narrows to a region in which features such as buildings, signals, or persons appear at high frequency so that the client can later obtain a WLD of such narrowed region. This enables the client to obtain detailed information on a necessary region, while reducing the amount of data received from the server.

[0174]

The server may also create from a WLD different SWLDs for the respective objects, and the client may receive SWLDs in accordance with the intended use. This reduces the network bandwidth. For example, the server recognizes persons or cars in a WLD in advance, and creates a SWLD of persons and a SWLD of cars. The client, when wishing to obtain information on persons around the client, receives the SWLD of persons, and when wising to obtain information on cars, receives the SWLD of cars. Such types of SWLDs

may be distinguished by information (flag, or type, etc.) added to the header, etc.

[0175]

Next, the structure and the operation flow of the three-dimensional data encoding device (e.g., a server) according to the present embodiment will be described. FIG. 16 is a block diagram of three-dimensional data encoding device 400 according to the present embodiment. FIG. 17 is a flowchart of three-dimensional data encoding processes performed by three-dimensional data encoding device 400.

[0176]

Three-dimensional data encoding device 400 shown in FIG. 16 encodes input three-dimensional data 411, thereby generating encoded three-dimensional data 413 and encoded three-dimensional data 414, each being an encoded stream. Here, encoded three-dimensional data 413 is encoded three-dimensional data corresponding to a WLD, and encoded three-dimensional data 414 is encoded three-dimensional data corresponding to a SWLD. Such three-dimensional data encoding device 400 includes, obtainer 401, encoding region determiner 402, SWLD extractor 403, WLD encoder 404, and SWLD encoder 405.

[0177]

First, as FIG. 17 shows, obtainer 401 obtains input three-dimensional data 411, which is point group data in a three-dimensional space (S401).

[0178]

Next, encoding region determiner 402 determines a current spatial region for encoding on the basis of a spatial region in which the point cloud data is present (S402).

[0179]

Next, SWLD extractor 403 defines the current spatial region as a WLD, and calculates the feature from each VXL included in the WLD. Then, SWLD extractor 403 extracts VXLs having an amount of features greater than or equal to a predetermined threshold, defines the extracted VXLs as FVXLs, and adds such FVXLs to a SWLD, thereby generating extracted three-dimensional data 412 (S403). Stated differently, extracted

three-dimensional data 412 having an amount of features greater than or equal to the threshold is extracted from input three-dimensional data 411.

[0180]

Next, WLD encoder 404 encodes input three-dimensional data 411 corresponding to the WLD, thereby generating encoded three-dimensional data 413 corresponding to the WLD (S404). In so doing, WLD encoder 404 adds to the header of encoded three-dimensional data 413 information that distinguishes that such encoded three-dimensional data 413 is a stream including a WLD.

[0181]

SWLD encoder 405 encodes extracted three-dimensional data 412 corresponding to the SWLD, thereby generating encoded three-dimensional data 414 corresponding to the SWLD (S405). In so doing, SWLD encoder 405 adds to the header of encoded three-dimensional data 414 information that distinguishes that such encoded three-dimensional data 414 is a stream including a SWLD.

[0182]

Note that the process of generating encoded three-dimensional data 413 and the process of generating encoded three-dimensional data 414 may be performed in the reverse order. Also note that a part or all of these processes may be performed in parallel.

[0183]

A parameter "world_type" is defined, for example, as information added to each header of encoded three-dimensional data 413 and encoded three-dimensional data 414. world_type=0 indicates that a stream includes a WLD, and world_type=1 indicates that a stream includes a SWLD. An increased number of values may be further assigned to define a larger number of types, e.g., world_type=2. Also, one of encoded three-dimensional data 413 and encoded three-dimensional data 414 may include a specified flag. For example, encoded three-dimensional data 414 may be assigned with a flag indicating that such stream includes a SWLD. In such a case, the decoding device can distinguish whether such stream

is a stream including a WLD or a stream including a SWLD in accordance with the presence/absence of the flag.

[0184]

Also, an encoding method used by WLD encoder 404 to encode a WLD may be different from an encoding method used by SWLD encoder 405 to encode a SWLD.

[0185]

For example, data of a SWLD is decimated, and thus can have a lower correlation with the neighboring data than that of a WLD. For this reason, of intra prediction and inter prediction, inter prediction may be more preferentially performed in an encoding method used for a SWLD than in an encoding method used for a WLD.

[0186]

Also, an encoding method used for a SWLD and an encoding method used for a WLD may represent three-dimensional positions differently. For example, three-dimensional coordinates may be used to represent the three-dimensional positions of FVXLs in a SWLD and an octree described below may be used to represent three-dimensional positions in a WLD, and vice versa.

[0187]

Also, SWLD encoder 405 performs encoding in a manner that encoded three-dimensional data 414 of a SWLD has a smaller data size than the data size of encoded three-dimensional data 413 of a WLD. A SWLD can have a lower inter-data correlation, for example, than that of a WLD as described above. This can lead to a decreased encoding efficiency, and thus to encoded three-dimensional data 414 having a larger data size than the data size of encoded three-dimensional data 413 of a WLD. When the data size of the resulting encoded three-dimensional data 414 is larger than the data size of encoded three-dimensional data 413 of a WLD, SWLD encoder 405 performs encoding again to re-generate encoded three-dimensional data 414 having a reduced data size.

[0188]

For example, SWLD extractor 403 re-generates extracted three-dimensional data 412 having a reduced number of keypoints to be extracted, and SWLD encoder 405 encodes such extracted three-dimensional data 412. Alternatively, SWLD encoder 405 may perform more coarse quantization. More coarse quantization is achieved, for example, by rounding the data in the lowermost level in an octree structure described below.

[0189]

When failing to decrease the data size of encoded three-dimensional data 414 of the SWLD to smaller than the data size of encoded three-dimensional data 413 of the WLD, SWLD encoder 405 may not generate encoded three-dimensional data 414 of the SWLD. Alternatively, encoded three-dimensional data 413 of the WLD may be copied as encoded three-dimensional data 414 of the SWLD. Stated differently, encoded three-dimensional data 413 of the WLD may be used as it is as encoded three-dimensional data 414 of the SWLD.

[0190]

Next, the structure and the operation flow of the three-dimensional data decoding device (e.g., a client) according to the present embodiment will be described. FIG. 18 is a block diagram of three-dimensional data decoding device 500 according to the present embodiment. FIG. 19 is a flowchart of three-dimensional data decoding processes performed by three-dimensional data decoding device 500.

[0191]

Three-dimensional data decoding device 500 shown in FIG. 18 decodes encoded three-dimensional data 511, thereby generating decoded three-dimensional data 512 or decoded three-dimensional data 513. Encoded three-dimensional data 511 here is, for example, encoded three-dimensional data 413 or encoded three-dimensional data 414 generated by three-dimensional data encoding device 400.

[0192]

Such three-dimensional data decoding device 500 includes obtainer 501, header

analyzer 502, WLD decoder 503, and SWLD decoder 504.

[0193]

First, as FIG. 19 shows, obtainer 501 obtains encoded three-dimensional data 511 (S501). Next, header analyzer 502 analyzes the header of encoded three-dimensional data 511 to identify whether encoded three-dimensional data 511 is a stream including a WLD or a stream including a SWLD (S502). For example, the above-described parameter world_type is referred to in making such identification.

[0194]

When encoded three-dimensional data 511 is a stream including a WLD (Yes in S503), WLD decoder 503 decodes encoded three-dimensional data 511, thereby generating decoded three-dimensional data 512 of the WLD (S504). Meanwhile, when encoded three-dimensional data 511 is a stream including a SWLD (No in S503), SWLD decoder 504 decodes encoded three-dimensional data 511, thereby generating decoded three-dimensional data 513 of the SWLD (S505).

[0195]

Also, as in the case of the encoding device, a decoding method used by WLD decoder 503 to decode a WLD may be different from a decoding method used by SWLD decoder 504 to decode a SWLD. For example, of intra prediction and inter prediction, inter prediction may be more preferentially performed in a decoding method used for a SWLD than in a decoding method used for a WLD.

[0196]

Also, a decoding method used for a SWLD and a decoding method used for a WLD may represent three-dimensional positions differently. For example, three-dimensional coordinates may be used to represent the three-dimensional positions of FVXLs in a SWLD and an octree described below may be used to represent three-dimensional positions in a WLD, and vice versa.

[0197]

Next, an octree representation will be described, which is a method of representing three-dimensional positions. VXL data included in three-dimensional data is converted into an octree structure before encoded. FIG. 20 is a diagram showing example VXLs in a WLD. FIG. 21 is a diagram showing an octree structure of the WLD shown in FIG. 20. An example shown in FIG. 20 illustrates three VXLs 1 to 3 that include point groups (hereinafter referred to as effective VXLs). As FIG. 21 shows, the octree structure is made of nodes and leaves. Each node has a maximum of eight nodes or leaves. Each leaf has VXL information. Here, of the leaves shown in FIG. 21, leaf 1, leaf 2, and leaf 3 represent VXL1, VXL2, and VXL3 shown in FIG. 20, respectively.

[0198]

More specifically, each node and each leaf correspond to a three-dimensional position. Node 1 corresponds to the entire block shown in FIG. 20. The block that corresponds to node 1 is divided into eight blocks. Of these eight blocks, blocks including effective VXLs are set as nodes, while the other blocks are set as leaves. Each block that corresponds to a node is further divided into eight nodes or leaves. These processes are repeated by the number of times that is equal to the number of levels in the octree structure. All blocks in the lowermost level are set as leaves.

[0199]

FIG. 22 is a diagram showing an example SWLD generated from the WLD shown in FIG. 20. VXL1 and VXL2 shown in FIG. 20 are judged as FVXL1 and FVXL2 as a result of feature extraction, and thus are added to the SWLD. Meanwhile, VXL3 is not judged as a FVXL, and thus is not added to the SWLD. FIG. 23 is a diagram showing an octree structure of the SWLD shown in FIG. 22. In the octree structure shown in FIG. 23, leaf 3 corresponding to VXL3 shown in FIG. 21 is deleted. Consequently, node 3 shown in FIG. 21 has lost an effective VXL, and has changed to a leaf. As described above, a SWLD has a smaller number of leaves in general than a WLD does, and thus the encoded three-dimensional data of the SWLD is smaller than the encoded three-dimensional data of

the WLD.

[0200]

The following describes variations of the present embodiment.

[0201]

For self-location estimation, for example, a client, being a vehicle-mounted device, etc., may receive a SWLD from the server to use such SWLD to estimate the self-location. Meanwhile, for obstacle detection, the client may detect obstacles by use of three-dimensional information on the periphery obtained by such client through various means including a distance sensor such as a rangefinder, as well as a stereo camera and a combination of a plurality of monocular cameras.

[0202]

In general, a SWLD is less likely to include VXL data on a flat region. As such, the server may hold a subsample world (subWLD) obtained by subsampling a WLD for detection of static obstacles, and send to the client the SWLD and the subWLD. This enables the client to perform self-location estimation and obstacle detection on the client's part, while reducing the network bandwidth.

[0203]

When the client renders three-dimensional map data at a high speed, map information having a mesh structure is more useful in some cases. As such, the server may generate a mesh from a WLD to hold it beforehand as a mesh world (MWLD). For example, when wishing to perform coarse three-dimensional rendering, the client receives a MWLD, and when wishing to perform detailed three-dimensional rendering, the client receives a WLD. This reduces the network bandwidth.

[0204]

In the above description, the server sets, as FVXLs, VXLs having an amount of features greater than or equal to the threshold, but the server may calculate FVXLs by a different method. For example, the server may judge that a VXL, a VLM, a SPC, or a GOS

that constitutes a signal, or an intersection, etc. as necessary for self-location estimation, driving assist, or self-driving, etc., and incorporate such VXL, VLM, SPC, or GOS into a SWLD as a FVXL, a FVLM, a FSPC, or a FGOS. Such judgment may be made manually. Also, FVXLs, etc. that have been set on the basis of an amount of features may be added to FVXLs, etc. obtained by the above method. Stated differently, SWLD extractor 403 may further extract, from input three-dimensional data 411, data corresponding to an object having a predetermined attribute as extracted three-dimensional data 412.

[0205]

Also, that a VXL, a VLM, a SPC, or a GOS is necessary for such intended usage may be labeled separately from the features. The server may separately hold, as an upper layer of a SWLD (e.g., a lane world), FVXLs of a signal or an intersection, etc. necessary for self-location estimation, driving assist, or self-driving, etc.

[0206]

The server may also add an attribute to VXLs in a WLD on a random access basis or on a predetermined unit basis. An attribute, for example, includes information indicating whether VXLs are necessary for self-location estimation, or information indicating whether VXLs are important as traffic information such as a signal, or an intersection, etc. An attribute may also include a correspondence between VXLs and features (intersection, or road, etc.) in lane information (geographic data files (GDF), etc.).

[0207]

A method as described below may be used to update a WLD or a SWLD.

[0208]

Update information indicating changes, etc. in a person, a roadwork, or a tree line (for trucks) is uploaded to the server as point groups or meta data. The server updates a WLD on the basis of such uploaded information, and then updates a SWLD by use of the updated WLD.

[0209]

The client, when detecting a mismatch between the three-dimensional information such client has generated at the time of self-location estimation and the three-dimensional information received from the server, may send to the server the three-dimensional information such client has generated, together with an update notification. In such a case, the server updates the SWLD by use of the WLD. When the SWLD is not to be updated, the server judges that the WLD itself is old.

[0210]

In the above description, information that distinguishes whether an encoded stream is that of a WLD or a SWLD is added as header information of the encoded stream. However, when there are many types of worlds such as a mesh world and a lane world, information that distinguishes these types of the worlds may be added to header information. Also, when there are many SWLDs with different amounts of features, information that distinguishes the respective SWLDs may be added to header information.

[0211]

In the above description, a SWLD is constituted by FVXLs, but a SWLD may include VXLs that have not been judged as FVXLs. For example, a SWLD may include an adjacent VXL used to calculate the feature of a FVXL. This enables the client to calculate the feature of a FVXL when receiving a SWLD, even in the case where feature information is not added to each FVXL of the SWLD. In such a case, the SWLD may include information that distinguishes whether each VXL is a FVXL or a VXL.

[0212]

As described above, three-dimensional data encoding device 400 extracts, from input three-dimensional data 411 (first three-dimensional data), extracted three-dimensional data 412 (second three-dimensional data) having an amount of a feature greater than or equal to a threshold, and encodes extracted three-dimensional data 412 to generate encoded three-dimensional data 414 (first encoded three-dimensional data).

[0213]

This three-dimensional data encoding device 400 generates encoded three-dimensional data 414 that is obtained by encoding data having an amount of a feature greater than or equal to the threshold. This reduces the amount of data compared to the case where input three-dimensional data 411 is encoded as it is. Three-dimensional data encoding device 400 is thus capable of reducing the amount of data to be transmitted.

[0214]

Three-dimensional data encoding device 400 further encodes input three-dimensional data 411 to generate encoded three-dimensional data 413 (second encoded three-dimensional data).

[0215]

This three-dimensional data encoding device 400 enables selective transmission of encoded three-dimensional data 413 and encoded three-dimensional data 414, in accordance, for example, with the intended use, etc.

[0216]

Also, extracted three-dimensional data 412 is encoded by a first encoding method, and input three-dimensional data 411 is encoded by a second encoding method different from the first encoding method.

[0217]

This three-dimensional data encoding device 400 enables the use of an encoding method suitable for each of input three-dimensional data 411 and extracted three-dimensional data 412.

[0218]

Also, of intra prediction and inter prediction, the inter prediction is more preferentially performed in the first encoding method than in the second encoding method.

[0219]

This three-dimensional data encoding device 400 enables inter prediction to be more preferentially performed on extracted three-dimensional data 412 in which adjacent data

items are likely to have low correlation.

[0220]

Also, the first encoding method and the second encoding method represent three-dimensional positions differently. For example, the second encoding method represents three-dimensional positions by octree, and the first encoding method represents three-dimensional positions by three-dimensional coordinates.

[0221]

This three-dimensional data encoding device 400 enables the use of a more suitable method to represent the three-dimensional positions of three-dimensional data in consideration of the difference in the number of data items (the number of VXLs or FVXLs) included.

[0222]

Also, at least one of encoded three-dimensional data 413 and encoded three-dimensional data 414 includes an identifier indicating whether the encoded three-dimensional data is encoded three-dimensional data obtained by encoding input three-dimensional data 411 or encoded three-dimensional data obtained by encoding part of input three-dimensional data 411. Stated differently, such identifier indicates whether the encoded three-dimensional data is encoded three-dimensional data 413 of a WLD or encoded three-dimensional data 414 of a SWLD.

[0223]

This enables the decoding device to readily judge whether the obtained encoded three-dimensional data is encoded three-dimensional data 413 or encoded three-dimensional data 414.

[0224]

Also, three-dimensional data encoding device 400 encodes extracted three-dimensional data 412 in a manner that encoded three-dimensional data 414 has a smaller data amount than a data amount of encoded three-dimensional data 413.

[0225]

This three-dimensional data encoding device 400 enables encoded three-dimensional data 414 to have a smaller data amount than the data amount of encoded three-dimensional data 413.

[0226]

Also, three-dimensional data encoding device 400 further extracts data corresponding to an object having a predetermined attribute from input three-dimensional data 411 as extracted three-dimensional data 412. The object having a predetermined attribute is, for example, an object necessary for self-location estimation, driving assist, or self-driving, etc., or more specifically, a signal, an intersection, etc.

[0227]

This three-dimensional data encoding device 400 is capable of generating encoded three-dimensional data 414 that includes data required by the decoding device.

[0228]

Also, three-dimensional data encoding device 400 (server) further sends, to a client, one of encoded three-dimensional data 413 and encoded three-dimensional data 414 in accordance with a status of the client.

[0229]

This three-dimensional data encoding device 400 is capable of sending appropriate data in accordance with the status of the client.

[0230]

Also, the status of the client includes one of a communication condition (e.g., network bandwidth) of the client and a traveling speed of the client.

[0231]

Also, three-dimensional data encoding device 400 further sends, to a client, one of encoded three-dimensional data 413 and encoded three-dimensional data 414 in accordance with a request from the client.

[0232]

This three-dimensional data encoding device 400 is capable of sending appropriate data in accordance with the request from the client.

[0233]

Also, three-dimensional data decoding device 500 according to the present embodiment decodes encoded three-dimensional data 413 or encoded three-dimensional data 414 generated by three-dimensional data encoding device 400 described above.

[0234]

Stated differently, three-dimensional data decoding device 500 decodes, by a first decoding method, encoded three-dimensional data 414 obtained by encoding extracted three-dimensional data 412 having an amount of a feature greater than or equal to a threshold, extracted three-dimensional data 412 having been extracted from input three-dimensional data 411. Three-dimensional data decoding device 500 also decodes, by a second decoding method, encoded three-dimensional data 413 obtained by encoding input three-dimensional data 411, the second decoding method being different from the first decoding method.

[0235]

This three-dimensional data decoding device 500 enables selective reception of encoded three-dimensional data 414 obtained by encoding data having an amount of a feature greater than or equal to the threshold and encoded three-dimensional data 413, in accordance, for example, with the intended use, etc. Three-dimensional data decoding device 500 is thus capable of reducing the amount of data to be transmitted. Such three-dimensional data decoding device 500 further enables the use of a decoding method suitable for each of input three-dimensional data 411 and extracted three-dimensional data 412.

[0236]

Also, of intra prediction and inter prediction, the inter prediction is more preferentially performed in the first decoding method than in the second decoding method.

[0237]

This three-dimensional data decoding device 500 enables inter prediction to be more preferentially performed on the extracted three-dimensional data in which adjacent data items are likely to have low correlation.

[0238]

Also, the first decoding method and the second decoding method represent three-dimensional positions differently. For example, the second decoding method represents three-dimensional positions by octree, and the first decoding method represents three-dimensional positions by three-dimensional coordinates.

[0239]

This three-dimensional data decoding device 500 enables the use of a more suitable method to represent the three-dimensional positions of three-dimensional data in consideration of the difference in the number of data items (the number of VXLs or FVXLs) included.

[0240]

Also, at least one of encoded three-dimensional data 413 and encoded three-dimensional data 414 includes an identifier indicating whether the encoded three-dimensional data is encoded three-dimensional data obtained by encoding input three-dimensional data 411 or encoded three-dimensional data obtained by encoding part of input three-dimensional data 411. Three-dimensional data decoding device 500 refers to such identifier in identifying between encoded three-dimensional data 413 and encoded three-dimensional data 414.

[0241]

This three-dimensional data decoding device 500 is capable of readily judging whether the obtained encoded three-dimensional data is encoded three-dimensional data 413 or encoded three-dimensional data 414.

[0242]

Three-dimensional data decoding device 500 further notifies a server of a status of

the client (three-dimensional data decoding device 500). Three-dimensional data decoding device 500 receives one of encoded three-dimensional data 413 and encoded three-dimensional data 414 from the server, in accordance with the status of the client.

[0243]

This three-dimensional data decoding device 500 is capable of receiving appropriate data in accordance with the status of the client.

[0244]

Also, the status of the client includes one of a communication condition (e.g., network bandwidth) of the client and a traveling speed of the client.

[0245]

Three-dimensional data decoding device 500 further makes a request of the server for one of encoded three-dimensional data 413 and encoded three-dimensional data 414, and receives one of encoded three-dimensional data 413 and encoded three-dimensional data 414 from the server, in accordance with the request.

[0246]

This three-dimensional data decoding device 500 is capable of receiving appropriate data in accordance with the intended use.

[0247]

EMBODIMENT 3

The present embodiment will describe a method of transmitting/receiving three-dimensional data between vehicles. For example, the three-dimensional data is transmitted/received between the own vehicle and the nearby vehicle.

[0248]

FIG. 24 is a block diagram of three-dimensional data creation device 620 according to the present embodiment. Such three-dimensional data creation device 620, which is included, for example, in the own vehicle, mergers first three-dimensional data 632 created by three-dimensional data creation device 620 with the received second three-dimensional

data 635, thereby creating third three-dimensional data 636 having a higher density.

[0249]

Such three-dimensional data creation device 620 includes three-dimensional data creator 621, request range determiner 622, searcher 623, receiver 624, decoder 625, and merger 626.

[0250]

First, three-dimensional data creator 621 creates first three-dimensional data 632 by use of sensor information 631 detected by the sensor included in the own vehicle. Next, request range determiner 622 determines a request range, which is the range of a three-dimensional space, the data on which is insufficient in the created first three-dimensional data 632.

[0251]

Next, searcher 623 searches for the nearby vehicle having the three-dimensional data of the request range, and sends request range information 633 indicating the request range to nearby vehicle 601 having been searched out (S623). Next, receiver 624 receives encoded three-dimensional data 634, which is an encoded stream of the request range, from nearby vehicle 601 (S624). Note that searcher 623 may indiscriminately send requests to all vehicles included in a specified range to receive encoded three-dimensional data 634 from a vehicle that has responded to the request. Searcher 623 may send a request not only to vehicles but also to an object such as a signal and a sign, and receive encoded three-dimensional data 634 from the object.

[0252]

Next, decoder 625 decodes the received encoded three-dimensional data 634, thereby obtaining second three-dimensional data 635. Next, merger 626 merges first three-dimensional data 632 with second three-dimensional data 635, thereby creating three-dimensional data 636 having a higher density.

[0253]

Next, the structure and operations of three-dimensional data transmission device 640 according to the present embodiment will be described. FIG. 25 is a block diagram of three-dimensional data transmission device 640.

[0254]

Three-dimensional data transmission device 640 is included, for example, in the above-described nearby vehicle. Three-dimensional data transmission device 640 processes fifth three-dimensional data 652 created by the nearby vehicle into sixth three-dimensional data 654 requested by the own vehicle, encodes sixth three-dimensional data 654 to generate encoded three-dimensional data 634, and sends encoded three-dimensional data 634 to the own vehicle.

[0255]

Three-dimensional data transmission device 640 includes three-dimensional data creator 641, receiver 642, extractor 643, encoder 644, and transmitter 645.

[0256]

First, three-dimensional data creator 641 creates fifth three-dimensional data 652 by use of sensor information 651 detected by the sensor included in the nearby vehicle. Next, receiver 642 receives request range information 633 from the own vehicle.

[0257]

Next, extractor 643 extracts from fifth three-dimensional data 652 the three-dimensional data of the request range indicated by request range information 633, thereby processing fifth three-dimensional data 652 into sixth three-dimensional data 654. Next, encoder 644 encodes sixth three-dimensional data 654 to generate encoded three-dimensional data 643, which is an encoded stream. Then, transmitter 645 sends encoded three-dimensional data 634 to the own vehicle.

[0258]

Note that although an example case is described here in which the own vehicle includes three-dimensional data creation device 620 and the nearby vehicle includes

three-dimensional data transmission device 640, each of the vehicles may include the functionality of both three-dimensional data creation device 620 and three-dimensional data transmission device 640.

[0259]

EMBODIMENT 4

The present embodiment describes operations performed in abnormal cases when self-location estimation is performed on the basis of a three-dimensional map.

[0260]

A three-dimensional map is expected to find its expanded use in self-driving of a vehicle and autonomous movement, etc. of a mobile object such as a robot and a flying object (e.g., a drone). Example means for enabling such autonomous movement include a method in which a mobile object travels in accordance with a three-dimensional map, while estimating its self-location on the map (self-location estimation).

[0261]

The self-location estimation is enabled by matching a three-dimensional map with three-dimensional information on the surrounding of the own vehicle (hereinafter referred to as self-detected three-dimensional data) obtained by a sensor equipped in the own vehicle, such as a rangefinder (e.g., a LiDAR) and a stereo camera to estimate the location of the own vehicle on the three-dimensional map.

[0262]

As in the case of an HD map suggested by HERE Technologies, for example, a three-dimensional map may include not only a three-dimensional point cloud, but also two-dimensional map data such as information on the shapes of roads and intersections, or information that changes in real-time such as information on a traffic jam and an accident. A three-dimensional map includes a plurality of layers such as layers of three-dimensional data, two-dimensional data, and meta-data that changes in real-time, from among which the device can obtain or refer to only necessary data.

[0263]

Point cloud data may be a SWLD as described above, or may include point group data that is different from keypoints. The transmission/reception of point cloud data is basically carried out in one or more random access units.

[0264]

A method described below is used as a method of matching a three-dimensional map with self-detected three-dimensional data. For example, the device compares the shapes of the point groups in each other's point clouds, and determines that portions having a high degree of similarity among keypoints correspond to the same position. When the three-dimensional map is formed by a SWLD, the device also performs matching by comparing the keypoints that form the SWLD with three-dimensional keypoints extracted from the self-detected three-dimensional data.

[0265]

Here, to enable highly accurate self-location estimation, the following needs to be satisfied: (A) the three-dimensional map and the self-detected three-dimensional data have been already obtained; and (B) their accuracies satisfy a predetermined requirement. However, one of (A) and (B) cannot be satisfied in abnormal cases such as ones described below.

[0266]

1. A three-dimensional map is unobtainable over communication.

[0267]

2. A three-dimensional map is not present, or a three-dimensional map having been obtained is corrupt.

[0268]

3. A sensor of the own vehicle has trouble, or the accuracy of the generated self-detected three-dimensional data is inadequate due to bad weather.

[0269]

The following describes operations to cope with such abnormal cases. The following description illustrates an example case of a vehicle, but the method described below is applicable to mobile objects on the whole that are capable of autonomous movement, such as a robot and a drone.

[0270]

The following describes the structure of the three-dimensional information processing device and its operation according to the present embodiment capable of coping with abnormal cases regarding a three-dimensional map or self-detected three-dimensional data. FIG. 26 is a block diagram of an example structure of three-dimensional information processing device 700 according to the present embodiment.

[0271]

Three-dimensional information processing device 700 is equipped, for example, in a mobile object such as a car. As shown in FIG. 26, three-dimensional information processing device 700 includes three-dimensional map obtainer 701, self-detected data obtainer 702, abnormal case judgment unit 703, coping operation determiner 704, and operation controller 705.

[0272]

Note that three-dimensional information processing device 700 may include a non-illustrated two-dimensional or one-dimensional sensor that detects a structural object or a mobile object around the own vehicle, such as a camera capable of obtaining two-dimensional images and a sensor for one-dimensional data utilizing ultrasonic or laser. Three-dimensional information processing device 700 may also include a non-illustrated communication unit that obtains a three-dimensional map over a mobile communication network, such as 4G and 5G, or via inter-vehicle communication or road-to-vehicle communication.

[0273]

Three-dimensional map obtainer 701 obtains three-dimensional map 711 of the

surroundings of the traveling route. For example, three-dimensional map obtainer 701 obtains three-dimensional map 711 over a mobile communication network, or via inter-vehicle communication or road-to-vehicle communication.

[0274]

Next, self-detected data obtainer 702 obtains self-detected three-dimensional data 712 on the basis of sensor information. For example, self-detected data obtainer 702 generates self-detected three-dimensional data 712 on the basis of the sensor information obtained by a sensor equipped in the own vehicle.

[0275]

Next, abnormal case judgment unit 703 conducts a predetermined check of at least one of obtained three-dimensional map 711 and self-detected three-dimensional data 712 to detect an abnormal case. Stated differently, abnormal case judgment unit 703 judges whether at least one of obtained three-dimensional map 711 and self-detected three-dimensional data 712 is abnormal.

[0276]

When the abnormal case is detected, coping operation determiner 704 determines a coping operation to cope with such abnormal case. Next, operation controller 705 controls the operation of each of the processing units necessary to perform the coping operation.

[0277]

Meanwhile, when no abnormal case is detected, three-dimensional information processing device 700 terminates the process.

[0278]

Also, three-dimensional information processing device 700 estimates the location of the vehicle equipped with three-dimensional information processing device 700, using three-dimensional map 711 and self-detected three-dimensional data 712. Next, three-dimensional information processing device 700 performs the automatic operation of the vehicle by use of the estimated location of the vehicle.

[0279]

As described above, three-dimensional information processing device 700 obtains, via a communication channel, map data (three-dimensional map 711) that includes first three-dimensional position information. The first three-dimensional position information includes, for example, a plurality of random access units, each of which is an assembly of at least one subspace and is individually decodable, the at least one subspace having three-dimensional coordinates information and serving as a unit in which each of the plurality of random access units is encoded. The first three-dimensional position information is, for example, data (SWLD) obtained by encoding keypoints, each of which has an amount of a three-dimensional feature greater than or equal to a predetermined threshold.

[0280]

Three-dimensional information processing device 700 also generates second three-dimensional position information (self-detected three-dimensional data 712) from information detected by a sensor. Three-dimensional information processing device 700 then judges whether one of the first three-dimensional position information and the second three-dimensional position information is abnormal by performing, on one of the first three-dimensional position information and the second three-dimensional position information, a process of judging whether an abnormality is present.

[0281]

Three-dimensional information processing device 700 determines a coping operation to cope with the abnormality when one of the first three-dimensional position information and the second three-dimensional position information is judged to be abnormal. Three-dimensional information processing device 700 then executes a control that is required to perform the coping operation.

[0282]

This structure enables three-dimensional information processing device 700 to detect an abnormality regarding one of the first three-dimensional position information and the

second three-dimensional position information, and to perform a coping operation therefor.

[0283]

EMBODIMENT 5

The present embodiment describes a method, etc. of transmitting three-dimensional data to a following vehicle.

[0284]

FIG. 27 is a block diagram of an exemplary structure of three-dimensional data creation device 810 according to the present embodiment. Such three-dimensional data creation device 810 is equipped, for example, in a vehicle. Three-dimensional data creation device 810 transmits and receives three-dimensional data to and from an external cloud-based traffic monitoring system, a preceding vehicle, or a following vehicle, and creates and stores three-dimensional data.

[0285]

Three-dimensional data creation device 810 includes data receiver 811, communication unit 812, reception controller 813, format converter 814, a plurality of sensors 815, three-dimensional data creator 816, three-dimensional data synthesizer 817, three-dimensional data storage 818, communication unit 819, transmission controller 820, format converter 821, and data transmitter 822.

[0286]

Data receiver 811 receives three-dimensional data 831 from a cloud-based traffic monitoring system or a preceding vehicle. Three-dimensional data 831 includes, for example, information on a region undetectable by sensors 815 of the own vehicle, such as a point cloud, visible light video, depth information, sensor position information, and speed information.

[0287]

Communication unit 812 communicates with the cloud-based traffic monitoring system or the preceding vehicle to transmit a data transmission request, etc. to the

cloud-based traffic monitoring system or the preceding vehicle.

[0288]

Reception controller 813 exchanges information, such as information on supported formats, with a communications partner via communication unit 812 to establish communication with the communications partner.

[0289]

Format converter 814 applies format conversion, etc. on three-dimensional data 831 received by data receiver 811 to generate three-dimensional data 832. Format converter 814 also decompresses or decodes three-dimensional data 831 when three-dimensional data 831 is compressed or encoded.

[0290]

A plurality of sensors 815 are a group of sensors, such as visible light cameras and infrared cameras, that obtain information on the outside of the vehicle and generate sensor information 833. Sensor information 833 is, for example, three-dimensional data such as a point cloud (point group data), when sensors 815 are laser sensors such as LIDARs. Note that a single sensor may serve as a plurality of sensors 815.

[0291]

Three-dimensional data creator 816 generates three-dimensional data 834 from sensor information 833. Three-dimensional data 834 includes, for example, information such as a point cloud, visible light video, depth information, sensor position information, and speed information.

[0292]

Three-dimensional data synthesizer 817 synthesizes three-dimensional data 834 created on the basis of sensor information 833 of the own vehicle with three-dimensional data 832 created by the cloud-based traffic monitoring system or the preceding vehicle, etc., thereby forming three-dimensional data 835 of a space that includes the space ahead of the preceding vehicle undetectable by sensors 815 of the own vehicle.

[0293]

Three-dimensional data storage 818 stores generated three-dimensional data 835, etc.

[0294]

Communication unit 819 communicates with the cloud-based traffic monitoring system or the following vehicle to transmit a data transmission request, etc. to the cloud-based traffic monitoring system or the following vehicle.

[0295]

Transmission controller 820 exchanges information such as information on supported formats with a communications partner via communication unit 819 to establish communication with the communications partner. Transmission controller 820 also determines a transmission region, which is a space of the three-dimensional data to be transmitted, on the basis of three-dimensional data formation information on three-dimensional data 832 generated by three-dimensional data synthesizer 817 and the data transmission request from the communications partner.

[0296]

More specifically, transmission controller 820 determines a transmission region that includes the space ahead of the own vehicle undetectable by a sensor of the following vehicle, in response to the data transmission request from the cloud-based traffic monitoring system or the following vehicle. Transmission controller 820 judges, for example, whether a space is transmittable or whether the already transmitted space includes an update, on the basis of the three-dimensional data formation information to determine a transmission region. For example, transmission controller 820 determines, as a transmission region, a region that is: a region specified by the data transmission request; and a region, corresponding three-dimensional data 835 of which is present. Transmission controller 820 then notifies format converter 821 of the format supported by the communications partner and the transmission region.

[0297]

Of three-dimensional data 835 stored in three-dimensional data storage 818, format converter 821 converts three-dimensional data 836 of the transmission region into the format supported by the receiver end to generate three-dimensional data 837. Note that format converter 821 may compress or encode three-dimensional data 837 to reduce the data amount.

[0298]

Data transmitter 822 transmits three-dimensional data 837 to the cloud-based traffic monitoring system or the following vehicle. Such three-dimensional data 837 includes, for example, information on a blind spot, which is a region hidden from view of the following vehicle, such as a point cloud ahead of the own vehicle, visible light video, depth information, and sensor position information.

[0299]

Note that an example has been described in which format converter 814 and format converter 821 perform format conversion, etc., but format conversion may not be performed.

[0300]

With the above structure, three-dimensional data creation device 810 obtains, from an external device, three-dimensional data 831 of a region undetectable by sensors 815 of the own vehicle, and synthesizes three-dimensional data 831 with three-dimensional data 834 that is based on sensor information 833 detected by sensors 815 of the own vehicle, thereby generating three-dimensional data 835. Three-dimensional data creation device 810 is thus capable of generating three-dimensional data of a range undetectable by sensors 815 of the own vehicle.

[0301]

Three-dimensional data creation device 810 is also capable of transmitting, to the cloud-based traffic monitoring system or the following vehicle, etc., three-dimensional data of a space that includes the space ahead of the own vehicle undetectable by a sensor of the

following vehicle, in response to the data transmission request from the cloud-based traffic monitoring system or the following vehicle.

[0302]

EMBODIMENT 6

In embodiment 5, an example is described in which a client device of a vehicle or the like transmits three-dimensional data to another vehicle or a server such as a cloud-based traffic monitoring system. In the present embodiment, a client device transmits sensor information obtained through a sensor to a server or a client device.

[0303]

A structure of a system according to the present embodiment will first be described. FIG. 28 is a diagram showing the structure of a transmission/reception system of a three-dimensional map and sensor information according to the present embodiment. This system includes server 901, and client devices 902A and 902B. Note that client devices 902A and 902B are also referred to as client device 902 when no particular distinction is made therebetween.

[0304]

Client device 902 is, for example, a vehicle-mounted device equipped in a mobile object such as a vehicle. Server 901 is, for example, a cloud-based traffic monitoring system, and is capable of communicating with the plurality of client devices 902.

[0305]

Server 901 transmits the three-dimensional map formed by a point cloud to client device 902. Note that a structure of the three-dimensional map is not limited to a point cloud, and may also be another structure expressing three-dimensional data such as a mesh structure.

[0306]

Client device 902 transmits the sensor information obtained by client device 902 to server 901. The sensor information includes, for example, at least one of information

obtained by LIDAR, a visible light image, an infrared image, a depth image, sensor position information, or sensor speed information.

[0307]

The data to be transmitted and received between server 901 and client device 902 may be compressed in order to reduce data volume, and may also be transmitted uncompressed in order to maintain data precision. When compressing the data, it is possible to use a three-dimensional compression method on the point cloud based on, for example, an octree structure. It is possible to use a two-dimensional image compression method on the visible light image, the infrared image, and the depth image. The two-dimensional image compression method is, for example, MPEG-4 AVC or HEVC standardized by MPEG.

[0308]

Server 901 transmits the three-dimensional map managed by server 901 to client device 902 in response to a transmission request for the three-dimensional map from client device 902. Note that server 901 may also transmit the three-dimensional map without waiting for the transmission request for the three-dimensional map from client device 902. For example, server 901 may broadcast the three-dimensional map to at least one client device 902 located in a predetermined space. Server 901 may also transmit the three-dimensional map suited to a position of client device 902 at fixed time intervals to client device 902 that has received the transmission request once. Server 901 may also transmit the three-dimensional map managed by server 901 to client device 902 every time the three-dimensional map is updated.

[0309]

Client device 902 sends the transmission request for the three-dimensional map to server 901. For example, when client device 902 wants to perform the self-location estimation during traveling, client device 902 transmits the transmission request for the three-dimensional map to server 901.

[0310]

Note that in the following cases, client device 902 may send the transmission request for the three-dimensional map to server 901. Client device 902 may send the transmission request for the three-dimensional map to server 901 when the three-dimensional map stored by client device 902 is old. For example, client device 902 may send the transmission request for the three-dimensional map to server 901 when a fixed period has passed since the three-dimensional map is obtained by client device 902.

[0311]

Client device 902 may also send the transmission request for the three-dimensional map to server 901 before a fixed time when client device 902 exits a space shown in the three-dimensional map stored by client device 902. For example, client device 902 may send the transmission request for the three-dimensional map to server 901 when client device 902 is located within a predetermined distance from a boundary of the space shown in the three-dimensional map stored by client device 902. When a movement path and a movement speed of client device 902 are understood, a time when client device 902 exits the space shown in the three-dimensional map stored by client device 902 may be predicted based on the movement path and the movement speed of client device 902.

[0312]

Client device 902 may also send the transmission request for the three-dimensional map to server 901 when an error during alignment of the three-dimensional data and the three-dimensional map created from the sensor information by client device 902 is at least at a fixed level.

[0313]

Client device 902 transmits the sensor information to server 901 in response to a transmission request for the sensor information from server 901. Note that client device 902 may transmit the sensor information to server 901 without waiting for the transmission request for the sensor information from server 901. For example, client device 902 may periodically transmit the sensor information during a fixed period when client device 902 has

received the transmission request for the sensor information from server 901 once. Client device 902 may determine that there is a possibility of a change in the three-dimensional map of a surrounding area of client device 902 having occurred, and transmit this information and the sensor information to server 901, when the error during alignment of the three-dimensional data created by client device 902 based on the sensor information and the three-dimensional map obtained from server 901 is at least at the fixed level.

[0314]

Server 901 sends a transmission request for the sensor information to client device 902. For example, server 901 receives position information, such as GPS information, about client device 902 from client device 902. Server 901 sends the transmission request for the sensor information to client device 902 in order to generate a new three-dimensional map, when it is determined that client device 902 is approaching a space in which the three-dimensional map managed by server 901 contains little information, based on the position information about client device 902. Server 901 may also send the transmission request for the sensor information, when wanting to (i) update the three-dimensional map, (ii) check road conditions during snowfall, a disaster, or the like, or (iii) check traffic congestion conditions, accident/incident conditions, or the like.

[0315]

Client device 902 may set an amount of data of the sensor information to be transmitted to server 901 in accordance with communication conditions or bandwidth during reception of the transmission request for the sensor information to be received from server 901. Setting the amount of data of the sensor information to be transmitted to server 901 is, for example, increasing/reducing the data itself or appropriately selecting a compression method.

[0316]

FIG. 29 is a block diagram showing an example structure of client device 902. Client device 902 receives the three-dimensional map formed by a point cloud and the like

from server 901, and estimates a self-location of client device 902 using the three-dimensional map created based on the sensor information of client device 902. Client device 902 transmits the obtained sensor information to server 901.

[0317]

Client device 902 includes data receiver 1011, communication unit 1012, reception controller 1013, format converter 1014, sensors 1015, three-dimensional data creator 1016, three-dimensional image processor 1017, three-dimensional data storage 1018, format converter 1019, communication unit 1020, transmission controller 1021, and data transmitter 1022.

[0318]

Data receiver 1011 receives three-dimensional map 1031 from server 901. Three-dimensional map 1031 is data that includes a point cloud such as a WLD or a SWLD. Three-dimensional map 1031 may include compressed data or uncompressed data.

[0319]

Communication unit 1012 communicates with server 901 and transmits a data transmission request (e.g. transmission request for three-dimensional map) to server 901.

[0320]

Reception controller 1013 exchanges information, such as information on supported formats, with a communications partner via communication unit 1012 to establish communication with the communications partner.

[0321]

Format converter 1014 performs a format conversion and the like on three-dimensional map 1031 received by data receiver 1011 to generate three-dimensional map 1032. Format converter 1014 also performs a decompression or decoding process when three-dimensional map 1031 is compressed or encoded. Note that format converter 1014 does not perform the decompression or decoding process when three-dimensional map 1031 is uncompressed data.

[0322]

Sensors 815 are a group of sensors, such as LIDARs, visible light cameras, infrared cameras, or depth sensors that obtain information about the outside of a vehicle equipped with client device 902, and generate sensor information 1033. Sensor information 1033 is, for example, three-dimensional data such as a point cloud (point group data) when sensors 1015 are laser sensors such as LIDARs. Note that a single sensor may serve as sensors 1015.

[0323]

Three-dimensional data creator 1016 generates three-dimensional data 1034 of a surrounding area of the own vehicle based on sensor information 1033. For example, three-dimensional data creator 1016 generates point cloud data with color information on the surrounding area of the own vehicle using information obtained by LIDAR and visible light video obtained by a visible light camera.

[0324]

Three-dimensional image processor 1017 performs a self-location estimation process and the like of the own vehicle, using (i) the received three-dimensional map 1032 such as a point cloud, and (ii) three-dimensional data 1034 of the surrounding area of the own vehicle generated using sensor information 1033. Note that three-dimensional image processor 1017 may generate three-dimensional data 1035 about the surroundings of the own vehicle by merging three-dimensional map 1032 and three-dimensional data 1034, and may perform the self-location estimation process using the created three-dimensional data 1035.

[0325]

Three-dimensional data storage 1018 stores three-dimensional map 1032, three-dimensional data 1034, three-dimensional data 1035, and the like.

[0326]

Format converter 1019 generates sensor information 1037 by converting sensor information 1033 to a format supported by a receiver end. Note that format converter 1019

may reduce the amount of data by compressing or encoding sensor information 1037. Format converter 1019 may omit this process when format conversion is not necessary. Format converter 1019 may also control the amount of data to be transmitted in accordance with a specified transmission range.

[0327]

Communication unit 1020 communicates with server 901 and receives a data transmission request (transmission request for sensor information) and the like from server 901.

[0328]

Transmission controller 1021 exchanges information, such as information on supported formats, with a communications partner via communication unit 1020 to establish communication with the communications partner.

[0329]

Data transmitter 1022 transmits sensor information 1037 to server 901. Sensor information 1037 includes, for example, information obtained through sensors 1015, such as information obtained by LIDAR, a luminance image obtained by a visible light camera, an infrared image obtained by an infrared camera, a depth image obtained by a depth sensor, sensor position information, and sensor speed information.

[0330]

A structure of server 901 will be described next. FIG. 30 is a block diagram showing an example structure of server 901. Server 901 transmits sensor information from client device 902 and creates three-dimensional data based on the received sensor information. Server 901 updates the three-dimensional map managed by server 901 using the created three-dimensional data. Server 901 transmits the updated three-dimensional map to client device 902 in response to a transmission request for the three-dimensional map from client device 902.

[0331]

Server 901 includes data receiver 1111, communication unit 1112, reception controller 1113, format converter 1114, three-dimensional data creator 1116, three-dimensional data merger 1117, three-dimensional data storage 1118, format converter 1119, communication unit 1120, transmission controller 1121, and data transmitter 1122.

[0332]

Data receiver 1111 receives sensor information 1037 from client device 902. Sensor information 1037 includes, for example, information obtained by LIDAR, a luminance image obtained by a visible light camera, an infrared image obtained by an infrared camera, a depth image obtained by a depth sensor, sensor position information, sensor speed information, and the like.

[0333]

Communication unit 1112 communicates with client device 902 and transmits a data transmission request (e.g. transmission request for sensor information) and the like to client device 902.

[0334]

Reception controller 1113 exchanges information, such as information on supported formats, with a communications partner via communication unit 1112 to establish communication with the communications partner.

[0335]

Format converter 1114 generates sensor information 1132 by performing a decompression or decoding process when the received sensor information 1037 is compressed or encoded. Note that format converter 1114 does not perform the decompression or decoding process when sensor information 1037 is uncompressed data.

[0336]

Three-dimensional data creator 1116 generates three-dimensional data 1134 of a surrounding area of client device 902 based on sensor information 1132. For example, three-dimensional data creator 1116 generates point cloud data with color information on the

surrounding area of client device 902 using information obtained by LIDAR and visible light video obtained by a visible light camera.

[0337]

Three-dimensional data merger 1117 updates three-dimensional map 1135 by merging three-dimensional data 1134 created based on sensor information 1132 with three-dimensional map 1135 managed by server 901.

[0338]

Three-dimensional data storage 1118 stores three-dimensional map 1135 and the like.

[0339]

Format converter 1119 generates three-dimensional map 1031 by converting three-dimensional map 1135 to a format supported by the receiver end. Note that format converter 1119 may reduce the amount of data by compressing or encoding three-dimensional map 1135. Format converter 1119 may omit this process when format conversion is not necessary. Format converter 1119 may also control the amount of data to be transmitted in accordance with a specified transmission range.

[0340]

Communication unit 1120 communicates with client device 902 and receives a data transmission request (transmission request for three-dimensional map) and the like from client device 902.

[0341]

Transmission controller 1121 exchanges information, such as information on supported formats, with a communications partner via communication unit 1120 to establish communication with the communications partner.

[0342]

Data transmitter 1122 transmits three-dimensional map 1031 to client device 902. Three-dimensional map 1031 is data that includes a point cloud such as a WLD or a SWLD.

Three-dimensional map 1031 may include one of compressed data and uncompressed data.

[0343]

An operational flow of client device 902 will be described next. FIG. 31 is a flowchart of an operation when client device 902 obtains the three-dimensional map.

[0344]

Client device 902 first requests server 901 to transmit the three-dimensional map (point cloud, etc.) (S1001). At this point, by also transmitting the position information about client device 902 obtained through GPS and the like, client device 902 may also request server 901 to transmit a three-dimensional map relating to this position information.

[0345]

Client device 902 next receives the three-dimensional map from server 901 (S1002). When the received three-dimensional map is compressed data, client device 902 decodes the received three-dimensional map and generates an uncompressed three-dimensional map (S1003).

[0346]

Client device 902 next creates three-dimensional data 1034 of the surrounding area of client device 902 using sensor information 1033 obtained by sensors 1015 (S1004). Client device 902 next estimates the self-location of client device 902 using three-dimensional map 1032 received from server 901 and three-dimensional data 1034 created using sensor information 1033 (S1005).

[0347]

FIG. 32 is a flowchart of an operation when client device 902 transmits the sensor information. Client device 902 first receives a transmission request for the sensor information from server 901 (S1011). Client device 902 that has received the transmission request transmits sensor information 1037 to server 901 (S1012). Note that client device 902 may generate sensor information 1037 by compressing each piece of information using a compression method suited to each piece of information, when sensor information 1033

includes a plurality of pieces of information obtained by sensors 1015.

[0348]

An operational flow of server 901 will be described next. FIG. 33 is a flowchart of an operation when server 901 obtains the sensor information. Server 901 first requests client device 902 to transmit the sensor information (S1021). Server 901 next receives sensor information 1037 transmitted from client device 902 in accordance with the request (S1022). Server 901 next creates three-dimensional data 1134 using the received sensor information 1037 (S1023). Server 901 next reflects the created three-dimensional data 1134 in three-dimensional map 1135 (S1024).

[0349]

FIG. 34 is a flowchart of an operation when server 901 transmits the three-dimensional map. Server 901 first receives a transmission request for the three-dimensional map from client device 902 (S1031). Server 901 that has received the transmission request for the three-dimensional map transmits the three-dimensional map to client device 902 (S1032). At this point, server 901 may extract a three-dimensional map of a vicinity of client device 902 along with the position information about client device 902, and transmit the extracted three-dimensional map. Server 901 may compress the three-dimensional map formed by a point cloud using, for example, an octree structure compression method, and transmit the compressed three-dimensional map.

[0350]

Hereinafter, variations of the present embodiment will be described.

[0351]

Server 901 creates three-dimensional data 1134 of a vicinity of a position of client device 902 using sensor information 1037 received from client device 902. Server 901 next calculates a difference between three-dimensional data 1134 and three-dimensional map 1135, by matching the created three-dimensional data 1134 with three-dimensional map 1135 of the same area managed by server 901. Server 901 determines that a type of anomaly has

occurred in the surrounding area of client device 902, when the difference is greater than or equal to a predetermined threshold. For example, it is conceivable that a large difference occurs between three-dimensional map 1135 managed by server 901 and three-dimensional data 1134 created based on sensor information 1037, when land subsidence and the like occurs due to a natural disaster such as an earthquake.

[0352]

Sensor information 1037 may include information indicating at least one of a sensor type, a sensor performance, and a sensor model number. Sensor information 1037 may also be appended with a class ID and the like in accordance with the sensor performance. For example, when sensor information 1037 is obtained by LIDAR, it is conceivable to assign identifiers to the sensor performance. A sensor capable of obtaining information with precision in units of several millimeters is class 1, a sensor capable of obtaining information with precision in units of several centimeters is class 2, and a sensor capable of obtaining information with precision in units of several meters is class 3. Server 901 may estimate sensor performance information and the like from a model number of client device 902. For example, when client device 902 is equipped in a vehicle, server 901 may determine sensor specification information from a type of the vehicle. In this case, server 901 may obtain information on the type of the vehicle in advance, and the information may also be included in the sensor information. Server 901 may change a degree of correction with respect to three-dimensional data 1134 created using sensor information 1037, using the obtained sensor information 1037. For example, when the sensor performance is high in precision (class 1), server 901 does not correct three-dimensional data 1134. When the sensor performance is low in precision (class 3), server 901 corrects three-dimensional data 1134 in accordance with the precision of the sensor. For example, server 901 increases the degree (intensity) of correction with a decrease in the precision of the sensor.

[0353]

Server 901 may simultaneously send the transmission request for the sensor

information to the plurality of client devices 902 in a certain space. Server 901 does not need to use all of the sensor information for creating three-dimensional data 1134 and may, for example, select sensor information to be used in accordance with the sensor performance, when having received a plurality of pieces of sensor information from the plurality of client devices 902. For example, when updating three-dimensional map 1135, server 901 may select high-precision sensor information (class 1) from among the received plurality of pieces of sensor information, and create three-dimensional data 1134 using the selected sensor information.

[0354]

Server 901 is not limited to only being a server such as a cloud-based traffic monitoring system, and may also be another (vehicle-mounted) client device. FIG. 35 is a diagram of a system structure in this case.

[0355]

For example, client device 902C sends a transmission request for sensor information to client device 902A located nearby, and obtains the sensor information from client device 902A. Client device 902C then creates three-dimensional data using the obtained sensor information of client device 902A, and updates a three-dimensional map of client device 902C. This enables client device 902C to generate a three-dimensional map of a space that can be obtained from client device 902A, and fully utilize the performance of client device 902C. For example, such a case is conceivable when client device 902C has high performance.

[0356]

In this case, client device 902A that has provided the sensor information is given rights to obtain the high-precision three-dimensional map generated by client device 902C. Client device 902A receives the high-precision three-dimensional map from client device 902C in accordance with these rights.

[0357]

Server 901 may send the transmission request for the sensor information to the plurality of client devices 902 (client device 902A and client device 902B) located nearby client device 902C. When a sensor of client device 902A or client device 902B has high performance, client device 902C is capable of creating the three-dimensional data using the sensor information obtained by this high-performance sensor.

[0358]

FIG. 36 is a block diagram showing a functionality structure of server 901 and client device 902. Server 901 includes, for example, three-dimensional map compression/decoding processor 1201 that compresses and decodes the three-dimensional map and sensor information compression/decoding processor 1202 that compresses and decodes the sensor information.

[0359]

Client device 902 includes three-dimensional map decoding processor 1211 and sensor information compression processor 1212. Three-dimensional map decoding processor 1211 receives encoded data of the compressed three-dimensional map, decodes the encoded data, and obtains the three-dimensional map. Sensor information compression processor 1212 compresses the sensor information itself instead of the three-dimensional data created using the obtained sensor information, and transmits the encoded data of the compressed sensor information to server 901. With this structure, client device 902 does not need to internally store a processor that performs a process for compressing the three-dimensional data of the three-dimensional map (point cloud, etc.), as long as client device 902 internally stores a processor that performs a process for decoding the three-dimensional map (point cloud, etc.). This makes it possible to limit costs, power consumption, and the like of client device 902.

[0360]

As stated above, client device 902 according to the present embodiment is equipped in the mobile object, and creates three-dimensional data 1034 of a surrounding area of the

mobile object using sensor information 1033 that is obtained through sensor 1015 equipped in the mobile object and indicates a surrounding condition of the mobile object. Client device 902 estimates a self-location of the mobile object using the created three-dimensional data 1034. Client device 902 transmits the obtained sensor information 1033 to server 901 or another mobile object.

[0361]

This enables client device 902 to transmit sensor information 1033 to server 901 or the like. This makes it possible to further reduce the amount of transmission data compared to when transmitting the three-dimensional data. Since there is no need for client device 902 to perform processes such as compressing or encoding the three-dimensional data, it is possible to reduce the processing amount of client device 902. As such, client device 902 is capable of reducing the amount of data to be transmitted or simplifying the structure of the device.

[0362]

Client device 902 further transmits the transmission request for the three-dimensional map to server 901 and receives three-dimensional map 1031 from server 901. In the estimating of the self-location, client device 902 estimates the self-location using three-dimensional data 1034 and three-dimensional map 1032.

[0363]

Sensor information 1034 includes at least one of information obtained by a laser sensor, a luminance image, an infrared image, a depth image, sensor position information, or sensor speed information.

[0364]

Sensor information 1033 includes information that indicates a performance of the sensor.

[0365]

Client device 902 encodes or compresses sensor information 1033, and in the

transmitting of the sensor information, transmits sensor information 1037 that has been encoded or compressed to server 901 or another mobile object 902. This enables client device 902 to reduce the amount of data to be transmitted.

[0366]

For example, client device 902 includes a processor and memory. The processor performs the above processes using the memory.

[0367]

Server 901 according to the present embodiment is capable of communicating with client device 902 equipped in the mobile object, and receives sensor information 1037 that is obtained through sensor 1015 equipped in the mobile object and indicates a surrounding condition of the mobile object. Server 901 creates three-dimensional data 1134 of a surrounding area of the mobile object using the received sensor information 1037.

[0368]

With this, server 901 creates three-dimensional data 1134 using sensor information 1037 transmitted from client device 902. This makes it possible to further reduce the amount of transmission data compared to when client device 902 transmits the three-dimensional data. Since there is no need for client device 902 to perform processes such as compressing or encoding the three-dimensional data, it is possible to reduce the processing amount of client device 902. As such, server 901 is capable of reducing the amount of data to be transmitted or simplifying the structure of the device.

[0369]

Server 901 further transmits a transmission request for the sensor information to client device 902.

[0370]

Server 901 further updates three-dimensional map 1135 using the created three-dimensional data 1134, and transmits three-dimensional map 1135 to client device 902 in response to the transmission request for three-dimensional map 1135 from client device

902.

[0371]

Sensor information 1037 includes at least one of information obtained by a laser sensor, a luminance image, an infrared image, a depth image, sensor position information, or sensor speed information.

[0372]

Sensor information 1037 includes information that indicates a performance of the sensor.

[0373]

Server 901 further corrects the three-dimensional data in accordance with the performance of the sensor. This enables the three-dimensional data creation method to improve the quality of the three-dimensional data.

[0374]

In the receiving of the sensor information, server 901 receives a plurality of pieces of sensor information 1037 received from a plurality of client devices 902, and selects sensor information 1037 to be used in the creating of three-dimensional data 1134, based on a plurality of pieces of information that each indicates the performance of the sensor included in the plurality of pieces of sensor information 1037. This enables server 901 to improve the quality of three-dimensional data 1134.

[0375]

Server 901 decodes or decompresses the received sensor information 1037, and creates three-dimensional data 1134 using sensor information 1132 that has been decoded or decompressed. This enables server 901 to reduce the amount of data to be transmitted.

[0376]

For example, server 901 includes a processor and memory. The processor performs the above processes using the memory.

[0377]

EMBODIMENT 7

In the present embodiment, three-dimensional data encoding and decoding methods using an inter prediction process will be described.

[0378]

FIG. 37 is a block diagram of three-dimensional data encoding device 1300 according to the present embodiment. This three-dimensional data encoding device 1300 generates an encoded bitstream (hereinafter, also simply referred to as bitstream) that is an encoded signal, by encoding three-dimensional data. As illustrated in FIG. 37, three-dimensional data encoding device 1300 includes divider 1301, subtractor 1302, transformer 1303, quantizer 1304, inverse quantizer 1305, inverse transformer 1306, adder 1307, reference volume memory 1308, intra predictor 1309, reference space memory 1310, inter predictor 1311, prediction controller 1312, and entropy encoder 1313.

[0379]

Divider 1301 divides a plurality of volumes (VLMs) that are encoding units of each space (SPC) included in the three-dimensional data. Divider 1301 makes an octree representation (make into an octree) of voxels in each volume. Note that divider 1301 may make the spaces into an octree representation with the spaces having the same size as the volumes. Divider 1301 may also append information (depth information, etc.) necessary for making the octree representation to a header and the like of a bitstream.

[0380]

Subtractor 1302 calculates a difference between a volume (encoding target volume) outputted by divider 1301 and a predicted volume generated through intra prediction or inter prediction, which will be described later, and outputs the calculated difference to transformer 1303 as a prediction residual. FIG. 38 is a diagram showing an example calculation of the prediction residual. Note that bit sequences of the encoding target volume and the predicted volume shown here are, for example, position information indicating positions of three-dimensional points included in the volumes.

[0381]

Hereinafter, a scan order of an octree representation and voxels will be described. A volume is encoded after being converted into an octree structure (made into an octree). The octree structure includes nodes and leaves. Each node has eight nodes or leaves, and each leaf has voxel (VXL) information. FIG. 39 is a diagram showing an example structure of a volume including voxels. FIG. 40 is a diagram showing an example of the volume shown in FIG. 39 having been converted into the octree structure. Among the leaves shown in FIG. 40, leaves 1, 2, and 3 respectively represent VXL 1, VXL 2, and VXL 3, and represent VXLs including a point group (hereinafter, active VXLs).

[0382]

An octree is represented by, for example, binary sequences of 1s and 0s. For example, when giving the nodes or the active VXLs a value of 1 and everything else a value of 0, each node and leaf is assigned with the binary sequence shown in FIG. 40. Thus, this binary sequence is scanned in accordance with a breadth-first or a depth-first scan order. For example, when scanning breadth-first, the binary sequence shown in A of FIG. 41 is obtained. When scanning depth-first, the binary sequence shown in B of FIG. 41 is obtained. The binary sequences obtained through this scanning are encoded through entropy encoding, which reduces an amount of information.

[0383]

Depth information in the octree representation will be described next. Depth in the octree representation is used in order to control up to how fine a granularity point cloud information included in a volume is stored. Upon setting a great depth, it is possible to reproduce the point cloud information to a more precise level, but an amount of data for representing the nodes and leaves increases. Upon setting a small depth, however, the amount of data decreases, but some information that the point cloud information originally held is lost, since pieces of point cloud information including different positions and different colors are now considered as pieces of point cloud information including the same position

and the same color.

[0384]

For example, FIG. 42 is a diagram showing an example in which the octree with a depth of 2 shown in FIG. 40 is represented with a depth of 1. The octree shown in FIG. 42 has a lower amount of data than the octree shown in FIG. 40. In other words, the binarized octree shown in FIG. 42 has a lower bit count than the octree shown in FIG. 40. Leaf 1 and leaf 2 shown in FIG. 40 are represented by leaf 1 shown in FIG. 41. In other words, the information on leaf 1 and leaf 2 being in different positions is lost.

[0385]

FIG. 43 is a diagram showing a volume corresponding to the octree shown in FIG. 42. VXL 1 and VXL 2 shown in FIG. 39 correspond to VXL 12 shown in FIG. 43. In this case, three-dimensional data encoding device 1300 generates color information of VXL 12 shown in FIG. 43 using color information of VXL 1 and VXL 2 shown in FIG. 39. For example, three-dimensional data encoding device 1300 calculates an average value, a median, a weighted average value, or the like of the color information of VXL 1 and VXL 2 as the color information of VXL 12. In this manner, three-dimensional data encoding device 1300 may control a reduction of the amount of data by changing the depth of the octree.

[0386]

Three-dimensional data encoding device 1300 may set the depth information of the octree to units of worlds, units of spaces, or units of volumes. In this case, three-dimensional data encoding device 1300 may append the depth information to header information of the world, header information of the space, or header information of the volume. In all worlds, spaces, and volumes associated with different times, the same value may be used as the depth information. In this case, three-dimensional data encoding device 1300 may append the depth information to header information managing the worlds associated with all times.

[0387]

When the color information is included in the voxels, transformer 1303 applies frequency transformation, e.g. orthogonal transformation, to a prediction residual of the color information of the voxels in the volume. For example, transformer 1303 creates a one-dimensional array by scanning the prediction residual in a certain scan order. Subsequently, transformer 1303 transforms the one-dimensional array to a frequency domain by applying one-dimensional orthogonal transformation to the created one-dimensional array. With this, when a value of the prediction residual in the volume is similar, a value of a low-frequency component increases and a value of a high-frequency component decreases. As such, it is possible to more efficiently reduce an encoding amount in quantizer 1304.

[0388]

Transformer 1303 does not need to use orthogonal transformation in one dimension, but may also use orthogonal transformation in two or more dimensions. For example, transformer 1303 maps the prediction residual to a two-dimensional array in a certain scan order, and applies two-dimensional orthogonal transformation to the obtained two-dimensional array. Transformer 1303 may select an orthogonal transformation method to be used from a plurality of orthogonal transformation methods. In this case, three-dimensional data encoding device 1300 appends, to the bitstream, information indicating which orthogonal transformation method is used. Transformer 1303 may select an orthogonal transformation method to be used from a plurality of orthogonal transformation methods in different dimensions. In this case, three-dimensional data encoding device 1300 appends, to the bitstream, in how many dimensions the orthogonal transformation method is used.

[0389]

For example, transformer 1303 matches the scan order of the prediction residual to a scan order (breadth-first, depth-first, or the like) in the octree in the volume. This makes it possible to reduce overhead, since information indicating the scan order of the prediction residual does not need to be appended to the bitstream. Transformer 1303 may apply a scan

order different from the scan order of the octree. In this case, three-dimensional data encoding device 1300 appends, to the bitstream, information indicating the scan order of the prediction residual. This enables three-dimensional data encoding device 1300 to efficiently encode the prediction residual. Three-dimensional data encoding device 1300 may append, to the bitstream, information (flag, etc.) indicating whether to apply the scan order of the octree, and may also append, to the bitstream, information indicating the scan order of the prediction residual when the scan order of the octree is not applied.

[0390]

Transformer 1303 does not only transform the prediction residual of the color information, and may also transform other attribute information included in the voxels. For example, transformer 1303 may transform and encode information, such as reflectance information, obtained when obtaining a point cloud through LIDAR and the like.

[0391]

Transformer 1303 may skip these processes when the spaces do not include attribute information such as color information. Three-dimensional data encoding device 1300 may append, to the bitstream, information (flag) indicating whether to skip the processes of transformer 1303.

[0392]

Quantizer 1304 generates a quantized coefficient by performing quantization using a quantization control parameter on a frequency component of the prediction residual generated by transformer 1303. With this, the amount of information is further reduced. The generated quantized coefficient is outputted to entropy encoder 1313. Quantizer 1304 may control the quantization control parameter in units of worlds, units of spaces, or units of volumes. In this case, three-dimensional data encoding device 1300 appends the quantization control parameter to each header information and the like. Quantizer 1304 may perform quantization control by changing a weight per frequency component of the prediction residual. For example, quantizer 1304 may precisely quantize a low-frequency

component and roughly quantize a high-frequency component. In this case, three-dimensional data encoding device 1300 may append, to a header, a parameter expressing a weight of each frequency component.

[0393]

Quantizer 1304 may skip these processes when the spaces do not include attribute information such as color information. Three-dimensional data encoding device 1300 may append, to the bitstream, information (flag) indicating whether to skip the processes of quantizer 1304.

[0394]

Inverse quantizer 1305 generates an inverse quantized coefficient of the prediction residual by performing inverse quantization on the quantized coefficient generated by quantizer 1304 using the quantization control parameter, and outputs the generated inverse quantized coefficient to inverse transformer 1306.

[0395]

Inverse transformer 1306 generates an inverse transformation-applied prediction residual by applying inverse transformation on the inverse quantized coefficient generated by inverse quantizer 1305. This inverse transformation-applied prediction residual does not need to completely coincide with the prediction residual outputted by transformer 1303, since the inverse transformation-applied prediction residual is a prediction residual that is generated after the quantization.

[0396]

Adder 1307 adds, to generate a reconstructed volume, (i) the inverse transformation-applied prediction residual generated by inverse transformer 1306 to (ii) a predicted volume that is generated through intra prediction or intra prediction, which will be described later, and is used to generate a pre-quantized prediction residual. This reconstructed volume is stored in reference volume memory 1308 or reference space memory 1310.

[0397]

Intra predictor 1309 generates a predicted volume of an encoding target volume using attribute information of a neighboring volume stored in reference volume memory 1308. The attribute information includes color information or a reflectance of the voxels. Intra predictor 1309 generates a predicted value of color information or a reflectance of the encoding target volume.

[0398]

FIG. 44 is a diagram for describing an operation of intra predictor 1309. For example, intra predictor 1309 generates the predicted volume of the encoding target volume (volume idx = 3) shown in FIG. 44, using a neighboring volume (volume idx = 0). Volume idx here is identifier information that is appended to a volume in a space, and a different value is assigned to each volume. An order of assigning volume idx may be the same as an encoding order, and may also be different from the encoding order. For example, intra predictor 1309 uses an average value of color information of voxels included in volume idx = 0, which is a neighboring volume, as the predicted value of the color information of the encoding target volume shown in FIG. 44. In this case, a prediction residual is generated by deducting the predicted value of the color information from the color information of each voxel included in the encoding target volume. The following processes are performed by transformer 1303 and subsequent processors with respect to this prediction residual. In this case, three-dimensional data encoding device 1300 appends, to the bitstream, neighboring volume information and prediction mode information. The neighboring volume information here is information indicating a neighboring volume used in the prediction, and indicates, for example, volume idx of the neighboring volume used in the prediction. The prediction mode information here indicates a mode used to generate the predicted volume. The mode is, for example, an average value mode in which the predicted value is generated using an average value of the voxels in the neighboring volume, or a median mode in which the predicted value is generated using the median of the voxels in the neighboring volume.

[0399]

Intra predictor 1309 may generate the predicted volume using a plurality of neighboring volumes. For example, in the structure shown in FIG. 44, intra predictor 1309 generates predicted volume 0 using a volume with volume idx = 0, and generates predicted volume 1 using a volume with volume idx = 1. Intra predictor 1309 then generates an average of predicted volume 0 and predicted volume 1 as a final predicted volume. In this case, three-dimensional data encoding device 1300 may append, to the bitstream, a plurality of volumes idx of a plurality of volumes used to generate the predicted volume.

[0400]

FIG. 45 is a diagram schematically showing the inter prediction process according to the present embodiment. Inter predictor 1311 encodes (inter predicts) a space (SPC) associated with certain time T_Cur using an encoded space associated with different time T_LX. In this case, inter predictor 1311 performs an encoding process by applying a rotation and translation process to the encoded space associated with different time T_LX.

[0401]

Three-dimensional data encoding device 1300 appends, to the bitstream, RT information relating to a rotation and translation process suited to the space associated with different time T_LX. Different time T_LX is, for example, time T_L0 before certain time T_Cur. At this point, three-dimensional data encoding device 1300 may append, to the bitstream, RT information RT_L0 relating to a rotation and translation process suited to a space associated with time T_L0.

[0402]

Alternatively, different time T_LX is, for example, time T_L1 after certain time T_Cur. At this point, three-dimensional data encoding device 1300 may append, to the bitstream, RT information RT_L1 relating to a rotation and translation process suited to a space associated with time T_L1.

[0403]

Alternatively, inter predictor 1311 encodes (bidirectional prediction) with reference to the spaces associated with time T_L0 and time T_L1 that differ from each other. In this case, three-dimensional data encoding device 1300 may append, to the bitstream, both RT information RT_L0 and RT information RT_L1 relating to the rotation and translation process suited to the spaces thereof.

[0404]

Note that T_L0 has been described as being before T_Cur and T_L1 as being after T_Cur, but are not necessarily limited thereto. For example, T_L0 and T_L1 may both be before T_Cur. T_L0 and T_L1 may also both be after T_Cur.

[0405]

Three-dimensional data encoding device 1300 may append, to the bitstream, RT information relating to a rotation and translation process suited to spaces associated with different times, when encoding with reference to each of the spaces. For example, three-dimensional data encoding device 1300 manages a plurality of encoded spaces to be referred to, using two reference lists (list L0 and list L1). When a first reference space in list L0 is L0R0, a second reference space in list L0 is L0R1, a first reference space in list L1 is L1R0, and a second reference space in list L1 is L1R1, three-dimensional data encoding device 1300 appends, to the bitstream, RT information RT_L0R0 of L0R0, RT information RT_L0R1 of L0R1, RT information RT_L1R0 of L1R0, and RT information RT_L1R1 of L1R1. For example, three-dimensional data encoding device 1300 appends these pieces of RT information to a header and the like of the bitstream.

[0406]

Three-dimensional data encoding device 1300 determines whether to apply rotation and translation per reference space, when encoding with reference to reference spaces associated with different times. In this case, three-dimensional data encoding device 1300 may append, to header information and the like of the bitstream, information (RT flag, etc.) indicating whether rotation and translation are applied per reference space. For example,

three-dimensional data encoding device 1300 calculates the RT information and an Iterative Closest Point (ICP) error value, using an ICP algorithm per reference space to be referred to from the encoding target space. Three-dimensional data encoding device 1300 determines that rotation and translation do not need to be performed and sets the RT flag to OFF, when the ICP error value is lower than or equal to a predetermined fixed value. In contrast, three-dimensional data encoding device 1300 sets the RT flag to ON and appends the RT information to the bitstream, when the ICP error value exceeds the above fixed value.

[0407]

FIG. 46 is a diagram showing an example syntax to be appended to a header of the RT information and the RT flag. Note that a bit count assigned to each syntax may be decided based on a range of this syntax. For example, when eight reference spaces are included in reference list L0, 3 bits may be assigned to MaxRefSpc_l0. The bit count to be assigned may be variable in accordance with a value each syntax can be, and may also be fixed regardless of the value each syntax can be. When the bit count to be assigned is fixed, three-dimensional data encoding device 1300 may append this fixed bit count to other header information.

[0408]

MaxRefSpc_l0 shown in FIG. 46 indicates a number of reference spaces included in reference list L0. RT_flag_l0[i] is an RT flag of reference space i in reference list L0. When RT_flag_l0[i] is 1, rotation and translation are applied to reference space i. When RT_flag_l0[i] is 0, rotation and translation are not applied to reference space i.

[0409]

R_l0[i] and T_l0[i] are RT information of reference space i in reference list L0. R_l0[i] is rotation information of reference space i in reference list L0. The rotation information indicates contents of the applied rotation process, and is, for example, a rotation matrix or a quaternion. T_l0[i] is translation information of reference space i in reference list L0. The translation information indicates contents of the applied translation process,

and is, for example, a translation vector.

[0410]

MaxRefSpc_l1 indicates a number of reference spaces included in reference list L1. RT_flag_l1[i] is an RT flag of reference space i in reference list L1. When RT_flag_l1[i] is 1, rotation and translation are applied to reference space i. When RT_flag_l1[i] is 0, rotation and translation are not applied to reference space i.

[0411]

R_l1[i] and T_l1[i] are RT information of reference space i in reference list L1. R_l1[i] is rotation information of reference space i in reference list L1. The rotation information indicates contents of the applied rotation process, and is, for example, a rotation matrix or a quaternion. T_l1[i] is translation information of reference space i in reference list L1. The translation information indicates contents of the applied translation process, and is, for example, a translation vector.

[0412]

Inter predictor 1311 generates the predicted volume of the encoding target volume using information on an encoded reference space stored in reference space memory 1310. As stated above, before generating the predicted volume of the encoding target volume, inter predictor 1311 calculates RT information at an encoding target space and a reference space using an ICP algorithm, in order to approach an overall positional relationship between the encoding target space and the reference space. Inter predictor 1311 then obtains reference space B by applying a rotation and translation process to the reference space using the calculated RT information. Subsequently, inter predictor 1311 generates the predicted volume of the encoding target volume in the encoding target space using information in reference space B. Three-dimensional data encoding device 1300 appends, to header information and the like of the encoding target space, the RT information used to obtain reference space B.

[0413]

In this manner, inter predictor 1311 is capable of improving precision of the predicted volume by generating the predicted volume using the information of the reference space, after approaching the overall positional relationship between the encoding target space and the reference space, by applying a rotation and translation process to the reference space. It is possible to reduce the encoding amount since it is possible to limit the prediction residual. Note that an example has been described in which ICP is performed using the encoding target space and the reference space, but is not necessarily limited thereto. For example, inter predictor 1311 may calculate the RT information by performing ICP using at least one of (i) an encoding target space in which a voxel or point cloud count is pruned, or (ii) a reference space in which a voxel or point cloud count is pruned, in order to reduce the processing amount.

[0414]

When the ICP error value obtained as a result of the ICP is smaller than a predetermined first threshold, i.e., when for example the positional relationship between the encoding target space and the reference space is similar, inter predictor 1311 determines that a rotation and translation process is not necessary, and the rotation and translation process does not need to be performed. In this case, three-dimensional data encoding device 1300 may control the overhead by not appending the RT information to the bitstream.

[0415]

When the ICP error value is greater than a predetermined second threshold, inter predictor 1311 determines that a shape change between the spaces is large, and intra prediction may be applied on all volumes of the encoding target space. Hereinafter, spaces to which intra prediction is applied will be referred to as intra spaces. The second threshold is greater than the above first threshold. The present embodiment is not limited to ICP, and any type of method may be used as long as the method calculates the RT information using two voxel sets or two point cloud sets.

[0416]

When attribute information, e.g. shape or color information, is included in the three-dimensional data, inter predictor 1311 searches, for example, a volume whose attribute information, e.g. shape or color information, is the most similar to the encoding target volume in the reference space, as the predicted volume of the encoding target volume in the encoding target space. This reference space is, for example, a reference space on which the above rotation and translation process has been performed. Inter predictor 1311 generates the predicted volume using the volume (reference volume) obtained through the search. FIG. 47 is a diagram for describing a generating operation of the predicted volume. When encoding the encoding target volume (volume idx = 0) shown in FIG. 47 using inter prediction, inter predictor 1311 searches a volume with a smallest prediction residual, which is the difference between the encoding target volume and the reference volume, while sequentially scanning the reference volume in the reference space. Inter predictor 1311 selects the volume with the smallest prediction residual as the predicted volume. The prediction residuals of the encoding target volume and the predicted volume are encoded through the processes performed by transformer 1303 and subsequent processors. The prediction residual here is a difference between the attribute information of the encoding target volume and the attribute information of the predicted volume. Three-dimensional data encoding device 1300 appends, to the header and the like of the bitstream, volume idx of the reference volume in the reference space, as the predicted volume.

[0417]

In the example shown in FIG. 47, the reference volume with volume idx = 4 of reference space L0R0 is selected as the predicted volume of the encoding target volume. The prediction residuals of the encoding target volume and the reference volume, and reference volume idx = 4 are then encoded and appended to the bitstream.

[0418]

Note that an example has been described in which the predicted volume of the attribute information is generated, but the same process may be applied to the predicted

volume of the position information.

[0419]

Prediction controller 1312 controls whether to encode the encoding target volume using intra prediction or inter prediction. A mode including intra prediction and inter prediction is referred to here as a prediction mode. For example, prediction controller 1312 calculates the prediction residual when the encoding target volume is predicted using intra prediction and the prediction residual when the encoding target volume is predicted using inter prediction as evaluation values, and selects the prediction mode whose evaluation value is smaller. Note that prediction controller 1312 may calculate an actual encoding amount by applying orthogonal transformation, quantization, and entropy encoding to the prediction residual of the intra prediction and the prediction residual of the inter prediction, and select a prediction mode using the calculated encoding amount as the evaluation value. Overhead information (reference volume idx information, etc.) aside from the prediction residual may be added to the evaluation value. Prediction controller 1312 may continuously select intra prediction when it has been decided in advance to encode the encoding target space using intra space.

[0420]

Entropy encoder 1313 generates an encoded signal (encoded bitstream) by variable-length encoding the quantized coefficient, which is an input from quantizer 1304. To be specific, entropy encoder 1313, for example, binarizes the quantized coefficient and arithmetically encodes the obtained binary signal.

[0421]

A three-dimensional data decoding device that decodes the encoded signal generated by three-dimensional data encoding device 1300 will be described next. FIG. 48 is a block diagram of three-dimensional data decoding device 1400 according to the present embodiment. This three-dimensional data decoding device 1400 includes entropy decoder 1401, inverse quantizer 1402, inverse transformer 1403, adder 1404, reference volume

memory 1405, intra predictor 1406, reference space memory 1407, inter predictor 1408, and prediction controller 1409.

[0422]

Entropy decoder 1401 variable-length decodes the encoded signal (encoded bitstream). For example, entropy decoder 1401 generates a binary signal by arithmetically decoding the encoded signal, and generates a quantized coefficient using the generated binary signal.

[0423]

Inverse quantizer 1402 generates an inverse quantized coefficient by inverse quantizing the quantized coefficient inputted from entropy decoder 1401, using a quantization parameter appended to the bitstream and the like.

[0424]

Inverse transformer 1403 generates a prediction residual by inverse transforming the inverse quantized coefficient inputted from inverse quantizer 1402. For example, inverse transformer 1403 generates the prediction residual by inverse orthogonally transforming the inverse quantized coefficient, based on information appended to the bitstream.

[0425]

Adder 1404 adds, to generate a reconstructed volume, (i) the prediction residual generated by inverse transformer 1403 to (ii) a predicted volume generated through intra prediction or intra prediction. This reconstructed volume is outputted as decoded three-dimensional data and is stored in reference volume memory 1405 or reference space memory 1407.

[0426]

Intra predictor 1406 generates a predicted volume through intra prediction using a reference volume in reference volume memory 1405 and information appended to the bitstream. To be specific, intra predictor 1406 obtains neighboring volume information (e.g. volume idx) appended to the bitstream and prediction mode information, and generates the

predicted volume through a mode indicated by the prediction mode information, using a neighboring volume indicated in the neighboring volume information. Note that the specifics of these processes are the same as the above-mentioned processes performed by intra predictor 1309, except for which information appended to the bitstream is used.

[0427]

Inter predictor 1408 generates a predicted volume through inter prediction using a reference space in reference space memory 1407 and information appended to the bitstream. To be specific, inter predictor 1408 applies a rotation and translation process to the reference space using the RT information per reference space appended to the bitstream, and generates the predicted volume using the rotated and translated reference space. Note that when an RT flag is present in the bitstream per reference space, inter predictor 1408 applies a rotation and translation process to the reference space in accordance with the RT flag. Note that the specifics of these processes are the same as the above-mentioned processes performed by inter predictor 1311, except for which information appended to the bitstream is used.

[0428]

Prediction controller 1409 controls whether to decode a decoding target volume using intra prediction or inter prediction. For example, prediction controller 1409 selects intra prediction or inter prediction in accordance with information that is appended to the bitstream and indicates the prediction mode to be used. Note that prediction controller 1409 may continuously select intra prediction when it has been decided in advance to decode the decoding target space using intra space.

[0429]

Hereinafter, variations of the present embodiment will be described. In the present embodiment, an example has been described in which rotation and translation is applied in units of spaces, but rotation and translation may also be applied in smaller units. For example, three-dimensional data encoding device 1300 may divide a space into subspaces, and apply rotation and translation in units of subspaces. In this case, three-dimensional data

encoding device 1300 generates RT information per subspace, and appends the generated RT information to a header and the like of the bitstream. Three-dimensional data encoding device 1300 may apply rotation and translation in units of volumes, which is an encoding unit. In this case, three-dimensional data encoding device 1300 generates RT information in units of encoded volumes, and appends the generated RT information to a header and the like of the bitstream. The above may also be combined. In other words, three-dimensional data encoding device 1300 may apply rotation and translation in large units and subsequently apply rotation and translation in small units. For example, three-dimensional data encoding device 1300 may apply rotation and translation in units of spaces, and may also apply different rotations and translations to each of a plurality of volumes included in the obtained spaces.

[0430]

In the present embodiment, an example has been described in which rotation and translation is applied to the reference space, but is not necessarily limited thereto. For example, three-dimensional data encoding device 1300 may apply a scaling process and change a size of the three-dimensional data. Three-dimensional data encoding device 1300 may also apply one or two of the rotation, translation, and scaling. When applying the processes in multiple stages and different units as stated above, a type of the processes applied in each unit may differ. For example, rotation and translation may be applied in units of spaces, and translation may be applied in units of volumes.

[0431]

Note that these variations are also applicable to three-dimensional data decoding device 1400.

[0432]

As stated above, three-dimensional data encoding device 1300 according to the present embodiment performs the following processes. FIG. 48 is a flowchart of the inter prediction process performed by three-dimensional data encoding device 1300.

[0433]

Three-dimensional data encoding device 1300 generates predicted position information (e.g. predicted volume) using position information on three-dimensional points included in three-dimensional reference data (e.g. reference space) associated with a time different from a time associated with current three-dimensional data (e.g. encoding target space) (S1301). To be specific, three-dimensional data encoding device 1300 generates the predicted position information by applying a rotation and translation process to the position information on the three-dimensional points included in the three-dimensional reference data.

[0434]

Note that three-dimensional data encoding device 1300 may perform a rotation and translation process using a first unit (e.g. spaces), and may perform the generating of the predicted position information using a second unit (e.g. volumes) that is smaller than the first unit. For example, three-dimensional data encoding device 1300 searches a volume among a plurality of volumes included in the rotated and translated reference space, whose position information differs the least from the position information of the encoding target volume included in the encoding target space. Note that three-dimensional data encoding device 1300 may perform the rotation and translation process, and the generating of the predicted position information in the same unit.

[0435]

Three-dimensional data encoding device 1300 may generate the predicted position information by applying (i) a first rotation and translation process to the position information on the three-dimensional points included in the three-dimensional reference data, and (ii) a second rotation and translation process to the position information on the three-dimensional points obtained through the first rotation and translation process, the first rotation and translation process using a first unit (e.g. spaces) and the second rotation and translation process using a second unit (e.g. volumes) that is smaller than the first unit.

[0436]

For example, as illustrated in FIG. 41, the position information on the three-dimensional points and the predicted position information is represented using an octree structure. For example, the position information on the three-dimensional points and the predicted position information is expressed in a scan order that prioritizes a breadth over a depth in the octree structure. For example, the position information on the three-dimensional points and the predicted position information is expressed in a scan order that prioritizes a depth over a breadth in the octree structure.

[0437]

As illustrated in FIG. 46, three-dimensional data encoding device 1300 encodes an RT flag that indicates whether to apply the rotation and translation process to the position information on the three-dimensional points included in the three-dimensional reference data. In other words, three-dimensional data encoding device 1300 generates the encoded signal (encoded bitstream) including the RT flag. Three-dimensional data encoding device 1300 encodes RT information that indicates contents of the rotation and translation process. In other words, three-dimensional data encoding device 1300 generates the encoded signal (encoded bitstream) including the RT information. Note that three-dimensional data encoding device 1300 may encode the RT information when the RT flag indicates to apply the rotation and translation process, and does not need to encode the RT information when the RT flag indicates not to apply the rotation and translation process.

[0438]

The three-dimensional data includes, for example, the position information on the three-dimensional points and the attribute information (color information, etc.) of each three-dimensional point. Three-dimensional data encoding device 1300 generates predicted attribute information using the attribute information of the three-dimensional points included in the three-dimensional reference data (S1302).

[0439]

Three-dimensional data encoding device 1300 next encodes the position information

on the three-dimensional points included in the current three-dimensional data, using the predicted position information. For example, as illustrated in FIG. 38, three-dimensional data encoding device 1300 calculates differential position information, the differential position information being a difference between the predicted position information and the position information on the three-dimensional points included in the current three-dimensional data (S1303).

[0440]

Three-dimensional data encoding device 1300 encodes the attribute information of the three-dimensional points included in the current three-dimensional data, using the predicted attribute information. For example, three-dimensional data encoding device 1300 calculates differential attribute information, the differential attribute information being a difference between the predicted attribute information and the attribute information on the three-dimensional points included in the current three-dimensional data (S1304). Three-dimensional data encoding device 1300 next performs transformation and quantization on the calculated differential attribute information (S1305).

[0441]

Lastly, three-dimensional data encoding device 1300 encodes (e.g. entropy encodes) the differential position information and the quantized differential attribute information (S1036). In other words, three-dimensional data encoding device 1300 generates the encoded signal (encoded bitstream) including the differential position information and the differential attribute information.

[0442]

Note that when the attribute information is not included in the three-dimensional data, three-dimensional data encoding device 1300 does not need to perform steps S1302, S1304, and S1305. Three-dimensional data encoding device 1300 may also perform only one of the encoding of the position information on the three-dimensional points and the encoding of the attribute information of the three-dimensional points.

[0443]

An order of the processes shown in FIG. 49 is merely an example and is not limited thereto. For example, since the processes with respect to the position information (S1301 and S1303) and the processes with respect to the attribute information (S1302, S1304, and S1305) are separate from one another, they may be performed in an order of choice, and a portion thereof may also be performed in parallel.

[0444]

With the above, three-dimensional data encoding device 1300 according to the present embodiment generates predicted position information using position information on three-dimensional points included in three-dimensional reference data associated with a time different from a time associated with current three-dimensional data; and encodes differential position information, which is a difference between the predicted position information and the position information on the three-dimensional points included in the current three-dimensional data. This makes it possible to improve encoding efficiency since it is possible to reduce the amount of data of the encoded signal.

[0445]

Three-dimensional data encoding device 1300 according to the present embodiment generates predicted attribute information using attribute information on three-dimensional points included in three-dimensional reference data; and encodes differential attribute information, which is a difference between the predicted attribute information and the attribute information on the three-dimensional points included in the current three-dimensional data. This makes it possible to improve encoding efficiency since it is possible to reduce the amount of data of the encoded signal.

[0446]

For example, three-dimensional data encoding device 1300 includes a processor and memory. The processor uses the memory to perform the above processes.

[0447]

FIG. 48 is a flowchart of the inter prediction process performed by three-dimensional data decoding device 1400.

[0448]

Three-dimensional data decoding device 1400 decodes (e.g. entropy decodes) the differential position information and the differential attribute information from the encoded signal (encoded bitstream) (S1401).

[0449]

Three-dimensional data decoding device 1400 decodes, from the encoded signal, an RT flag that indicates whether to apply the rotation and translation process to the position information on the three-dimensional points included in the three-dimensional reference data. Three-dimensional data decoding device 1400 encodes RT information that indicates contents of the rotation and translation process. Note that three-dimensional data decoding device 1400 may decode the RT information when the RT flag indicates to apply the rotation and translation process, and does not need to decode the RT information when the RT flag indicates not to apply the rotation and translation process.

[0450]

Three-dimensional data decoding device 1400 next performs inverse transformation and inverse quantization on the decoded differential attribute information (S1402).

[0451]

Three-dimensional data decoding device 1400 next generates predicted position information (e.g. predicted volume) using the position information on the three-dimensional points included in the three-dimensional reference data (e.g. reference space) associated with a time different from a time associated with the current three-dimensional data (e.g. decoding target space) (S1403). To be specific, three-dimensional data decoding device 1400 generates the predicted position information by applying a rotation and translation process to the position information on the three-dimensional points included in the three-dimensional reference data.

[0452]

More specifically, when the RT flag indicates to apply the rotation and translation process, three-dimensional data decoding device 1400 applies the rotation and translation process on the position information on the three-dimensional points included in the three-dimensional reference data indicated in the RT information. In contrast, when the RT flag indicates not to apply the rotation and translation process, three-dimensional data decoding device 1400 does not apply the rotation and translation process on the position information on the three-dimensional points included in the three-dimensional reference data.

[0453]

Note that three-dimensional data decoding device 1400 may perform the rotation and translation process using a first unit (e.g. spaces), and may perform the generating of the predicted position information using a second unit (e.g. volumes) that is smaller than the first unit. Note that three-dimensional data decoding device 1400 may perform the rotation and translation process, and the generating of the predicted position information in the same unit.

[0454]

Three-dimensional data decoding device 1400 may generate the predicted position information by applying (i) a first rotation and translation process to the position information on the three-dimensional points included in the three-dimensional reference data, and (ii) a second rotation and translation process to the position information on the three-dimensional points obtained through the first rotation and translation process, the first rotation and translation process using a first unit (e.g. spaces) and the second rotation and translation process using a second unit (e.g. volumes) that is smaller than the first unit.

[0455]

For example, as illustrated in FIG. 41, the position information on the three-dimensional points and the predicted position information is represented using an octree structure. For example, the position information on the three-dimensional points and the predicted position information is expressed in a scan order that prioritizes a breadth over a

depth in the octree structure. For example, the position information on the three-dimensional points and the predicted position information is expressed in a scan order that prioritizes a depth over a breadth in the octree structure.

[0456]

Three-dimensional data decoding device 1400 generates predicted attribute information using the attribute information of the three-dimensional points included in the three-dimensional reference data (S1404).

[0457]

Three-dimensional data decoding device 1400 next restores the position information on the three-dimensional points included in the current three-dimensional data, by decoding encoded position information included in an encoded signal, using the predicted position information. The encoded position information here is the differential position information. Three-dimensional data decoding device 1400 restores the position information on the three-dimensional points included in the current three-dimensional data, by adding the differential position information to the predicted position information (S1405).

[0458]

Three-dimensional data decoding device 1400 restores the attribute information of the three-dimensional points included in the current three-dimensional data, by decoding encoded attribute information included in an encoded signal, using the predicted attribute information. The encoded attribute information here is the differential position information. Three-dimensional data decoding device 1400 restores the attribute information on the three-dimensional points included in the current three-dimensional data, by adding the differential attribute information to the predicted attribute information (S1406).

[0459]

Note that when the attribute information is not included in the three-dimensional data, three-dimensional data decoding device 1400 does not need to perform steps S1402, S1404, and S1406. Three-dimensional data decoding device 1400 may also perform only

one of the decoding of the position information on the three-dimensional points and the decoding of the attribute information of the three-dimensional points.

[0460]

An order of the processes shown in FIG. 50 is merely an example and is not limited thereto. For example, since the processes with respect to the position information (S1403 and S1405) and the processes with respect to the attribute information (S1402, S1404, and S1406) are separate from one another, they may be performed in an order of choice, and a portion thereof may also be performed in parallel.

[0461]

EMBODIMENT 8

In the present embodiment, a representation means of three-dimensional points (point cloud) in encoding of three-dimensional data will be described.

[0462]

FIG. 51 is a block diagram showing a structure of a distribution system of three-dimensional data according to the present embodiment. The distribution system shown in FIG. 51 includes server 1501 and a plurality of clients 1502.

[0463]

Server 1501 includes storage 1511 and controller 1512. Storage 1511 stores encoded three-dimensional map 1513 that is encoded three-dimensional data.

[0464]

FIG. 52 is a diagram showing an example structure of a bitstream of encoded three-dimensional map 1513. The three-dimensional map is divided into a plurality of submaps and each submap is encoded. Each submap is appended with a random-access (RA) header including subcoordinate information. The subcoordinate information is used for improving encoding efficiency of the submap. This subcoordinate information indicates subcoordinates of the submap. The subcoordinates are coordinates of the submap having reference coordinates as reference. Note that the three-dimensional map including the

plurality of submaps is referred to as an overall map. Coordinates that are a reference in the overall map (e.g. origin) are referred to as the reference coordinates. In other words, the subcoordinates are the coordinates of the submap in a coordinate system of the overall map. In other words, the subcoordinates indicate an offset between the coordinate system of the overall map and a coordinate system of the submap. Coordinates in the coordinate system of the overall map having the reference coordinates as reference are referred to as overall coordinates. Coordinates in the coordinate system of the submap having the subcoordinates as reference are referred to as differential coordinates.

[0465]

Client 1502 transmits a message to server 1501. This message includes position information on client 1502. Controller 1512 included in server 1501 obtains a bitstream of a submap located closest to client 1502, based on the position information included in the received message. The bitstream of the submap includes the subcoordinate information and is transmitted to client 1502. Decoder 1521 included in client 1502 obtains overall coordinates of the submap having the reference coordinates as reference, using this subcoordinate information. Application 1522 included in client 1502 executes an application relating to a self-location, using the obtained overall coordinates of the submap.

[0466]

The submap indicates a partial area of the overall map. The subcoordinates are the coordinates in which the submap is located in a reference coordinate space of the overall map. For example, in an overall map called A, there is submap A called AA and submap B called AB. When a vehicle wants to consult a map of AA, decoding begins from submap A, and when the vehicle wants to consult a map of AB, decoding begins from submap B. The submap here is a random-access point. To be specific, A is Osaka Prefecture, AA is Osaka City, and AB is Takatsuki City.

[0467]

Each submap is transmitted along with the subcoordinate information to the client.

The subcoordinate information is included in header information of each submap, a transmission packet, or the like.

[0468]

The reference coordinates, which serve as a reference for the subcoordinate information of each submap, may be appended to header information of a space at a higher level than the submap, such as header information of the overall map.

[0469]

The submap may be formed by one space (SPC). The submap may also be formed by a plurality of SPCs.

[0470]

The submap may include a Group of Spaces (GOS). The submap may be formed by a world. For example, in a case where there are a plurality of objects in the submap, the submap is formed by a plurality of SPCs when assigning the plurality of objects to separate SPCs. The submap is formed by one SPC when assigning the plurality of objects to one SPC.

[0471]

An advantageous effect on encoding efficiency when using the subcoordinate information will be described next. FIG. 53 is a diagram for describing this advantageous effect. For example, a high bit count is necessary in order to encode three-dimensional point A, which is located far from the reference coordinates, shown in FIG. 53. A distance between the subcoordinates and three-dimensional point A is shorter than a distance between the reference coordinates and three-dimensional point A. As such, it is possible to improve encoding efficiency by encoding coordinates of three-dimensional point A having the subcoordinates as reference more than when encoding the coordinates of three-dimensional point A having the reference coordinates as reference. The bitstream of the submap includes the subcoordinate information. By transmitting the bitstream of the submap and the reference coordinates to a decoding end (client), it is possible to restore the overall

coordinates of the submap in the decoder end.

[0472]

FIG. 54 is a flowchart of processes performed by server 1501, which is a transmission end of the submap.

[0473]

Server 1501 first receives a message including position information on client 1502 from client 1502 (S1501). Controller 1512 obtains an encoded bitstream of the submap based on the position information on the client from storage 1511 (S1502). Server 1501 then transmits the encoded bitstream of the submap and the reference coordinates to client 1502 (S1503).

[0474]

FIG. 55 is a flowchart of processes performed by client 1502, which is a receiver end of the submap.

[0475]

Client 1502 first receives the encoded bitstream of the submap and the reference coordinates transmitted from server 1501 (S1511). Client 1502 next obtains the subcoordinate information of the submap by decoding the encoded bitstream (S1512). Client 1502 next restores the differential coordinates in the submap to the overall coordinates, using the reference coordinates and the subcoordinates (S1513).

[0476]

An example syntax of information relating to the submap will be described next. In the encoding of the submap, the three-dimensional data encoding device calculates the differential coordinates by subtracting the subcoordinates from the coordinates of each point cloud (three-dimensional points). The three-dimensional data encoding device then encodes the differential coordinates into the bitstream as a value of each point cloud. The encoding device encodes the subcoordinate information indicating the subcoordinates as the header information of the bitstream. This enables the three-dimensional data decoding device to

obtain overall coordinates of each point cloud. For example, the three-dimensional data encoding device is included in server 1501 and the three-dimensional data decoding device is included in client 1502.

[0477]

FIG. 56 is a diagram showing an example syntax of the submap. NumOfPoint shown in FIG. 56 indicates a total number of point clouds included in the submap. sub_coordinate_x, sub_coordinate_y, and sub_coordinate_z are the subcoordinate information. sub_coordinate_x indicates an x-coordinate of the subcoordinates. sub_coordinate_y indicates a y-coordinate of the subcoordinates. sub_coordinate_z indicates a z-coordinate of the subcoordinates.

[0478]

diff_x[i], diff_y[i], and diff_z[i] are differential coordinates of an i-th point cloud in the submap. diff_x[i] is a differential value between an x-coordinate of the i-th point cloud and the x-coordinate of the subcoordinates in the submap. diff_y[i] is a differential value between a y-coordinate of the i-th point cloud and the y-coordinate of the subcoordinates in the submap. diff_z[i] is a differential value between a z-coordinate of the i-th point cloud and the z-coordinate of the subcoordinates in the submap.

[0479]

The three-dimensional data decoding device decodes point_cloud[i]_x, point_cloud[i]_y, and point_cloud[i]_z, which are overall coordinates of the i-th point cloud, using the expression below. point_cloud[i]_x is an x-coordinate of the overall coordinates of the i-th point cloud. point_cloud[i]_y is a y-coordinate of the overall coordinates of the i-th point cloud. point_cloud[i]_z is a z-coordinate of the overall coordinates of the i-th point cloud.

[0480]

point_cloud[i]_x = sub_coordinate_x + diff_x[i]

point_cloud[i]_y = sub_coordinate_y + diff_y[i]

point_cloud[i]_z = sub_coordinate_z + diff_z[i]

[0481]

A switching process for applying octree encoding will be described next. The three-dimensional data encoding device selects, when encoding the submap, whether to encode each point cloud using an octree representation (hereinafter, referred to as octree encoding) or to encode the differential values from the subcoordinates (hereinafter, referred to as non-octree encoding). FIG. 57 is a diagram schematically showing this operation. For example, the three-dimensional data encoding device applies octree encoding to the submap, when the total number of point clouds in the submap is at least a predetermined threshold. The three-dimensional data encoding device applies non-octree encoding to the submap, when the total number of point clouds in the submap is lower than the predetermined threshold. This enables the three-dimensional data encoding device to improve encoding efficiency, since it is possible to appropriately select whether to use octree encoding or non-octree encoding, in accordance with a shape and density of objects included in the submap.

[0482]

The three-dimensional data encoding device appends, to a header and the like of the submap, information indicating whether octree encoding or non-octree encoding has been applied to the submap (hereinafter, referred to as octree encoding application information). This enables the three-dimensional data decoding device to identify whether the bitstream is obtained by octree encoding the submap or non-octree encoding the submap.

[0483]

The three-dimensional data encoding device may calculate encoding efficiency when applying octree encoding and encoding efficiency when applying non-octree encoding to the same point cloud, and apply an encoding method whose encoding efficiency is better to the submap.

[0484]

FIG. 58 is a diagram showing an example syntax of the submap when performing this switching. coding_type shown in FIG. 58 is information indicating the encoding type and is the above octree encoding application information. coding_type = 00 indicates that octree encoding has been applied. coding_type = 01 indicates that non-octree encoding has been applied. coding_type = 10 or 11 indicates that an encoding method and the like other than the above encoding methods has been applied.

[0485]

When the encoding type is non-octree encoding (non_octree), the submap includes NumOfPoint and the subcoordinate information (sub_coordinate_x, sub_coordinate_y, and sub_coordinate_z).

[0486]

When the encoding type is octree encoding (octree), the submap includes octree_info. octree_info is information necessary to the octree encoding and includes, for example, depth information.

[0487]

When the encoding type is non-octree encoding (non_octree), the submap includes the differential coordinates (diff_x[i], diff_y[i], and diff_z[i]).

[0488]

When the encoding type is octree encoding (octree), the submap includes octree_data, which is encoded data relating to the octree encoding.

[0489]

Note that an example has been described here in which an xyz coordinate system is used as the coordinate system of the point cloud, but a polar coordinate system may also be used.

[0490]

FIG. 59 is a flowchart of a three-dimensional data encoding process performed by the three-dimensional data encoding device. Three-dimensional data encoding device first

calculates a total number of point clouds in a current submap, which is the submap to be processed (S1521). The three-dimensional data encoding device next determines whether when the calculated total number of point clouds is at least a predetermined threshold (S1522).

[0491]

When the total number of point clouds is at least the predetermined threshold (YES in S1522), the three-dimensional data encoding device applies octree encoding to the current submap (S1523). The three-dimensional data encoding device appends, to a header of the bitstream, octree encoding application information indicating that octree encoding has been applied to the current submap (S1525).

[0492]

In contrast, when the total number of point clouds is lower than the predetermined threshold (NO in S1522), the three-dimensional data encoding device applies non-octree encoding to the current submap (S1524). The three-dimensional data encoding device appends, to the header of the bitstream, octree encoding application information indicating that non-octree encoding has been applied to the current submap (S1525).

[0493]

FIG. 60 is a flowchart of a three-dimensional data decoding process performed by the three-dimensional data decoding device. The three-dimensional data decoding device first decodes the octree encoding application information from the header of the bitstream (S1531). The three-dimensional data decoding device next determines whether the encoding type applied to the current submap is octree encoding, based on the decoded octree encoding application information (S1532).

[0494]

When the octree encoding application information indicates that the encoding type is octree encoding (YES in S1532), the three-dimensional data decoding device decodes the current submap through octree decoding (S1533). In contrast, when the octree encoding

application information indicates that the encoding type is non-octree encoding (NO in S1532), the three-dimensional data decoding device decodes the current submap through non-octree decoding (S1534).

[0495]

Hereinafter, variations of the present embodiment will be described. FIG. 61 to FIG. 63 are diagrams schematically showing operations of variations of the switching process of the encoding type.

[0496]

As illustrated in FIG. 61, the three-dimensional data encoding device may select whether to apply octree encoding or non-octree encoding per space. In this case, the three-dimensional data encoding device appends the octree encoding application information to a header of the space. This enables the three-dimensional data decoding device to determine whether octree encoding has been applied per space. In this case, the three-dimensional data encoding device sets subcoordinates per space, and encodes a differential value, which is a value of the subcoordinates subtracted from coordinates of each point cloud in the space.

[0497]

This enables the three-dimensional data encoding device to improve encoding efficiency, since it is possible to appropriately select whether to apply octree encoding, in accordance with a shape of objects or the total number of point clouds in the space.

[0498]

As illustrated in FIG. 62, the three-dimensional data encoding device may select whether to apply octree encoding or non-octree encoding per volume. In this case, the three-dimensional data encoding device appends the octree encoding application information to a header of the volume. This enables the three-dimensional data decoding device to determine whether octree encoding has been applied per volume. In this case, the three-dimensional data encoding device sets subcoordinates per volume, and encodes a

differential value, which is a value of the subcoordinates subtracted from coordinates of each point cloud in the volume.

[0499]

This enables the three-dimensional data encoding device to improve encoding efficiency, since it is possible to appropriately select whether to apply octree encoding, in accordance with a shape of objects or the total number of point clouds in the volume.

[0500]

In the above description, an example has been shown in which the difference, which is the subcoordinates of each point cloud subtracted from the coordinates of each point cloud, is encoded as the non-octree encoding, but is not limited thereto, and any other type of encoding method other than the octree encoding may be used. For example, as illustrated in FIG. 63, the three-dimensional data encoding device may not only encode the difference from the subcoordinates as the non-octree encoding, but also use a method in which a value of the point cloud in the submap, the space, or the volume itself is encoded (hereinafter, referred to as original coordinate encoding).

[0501]

In this case, the three-dimensional data encoding device stores, in the header, information indicating that original coordinate encoding has been applied to a current space (submap, space, or volume). This enables the three-dimensional data decoding device to determine whether original coordinate encoding has been applied to the current space.

[0502]

When applying original coordinate encoding, the three-dimensional data encoding device may perform the encoding without applying quantization and arithmetic encoding to original coordinates. The three-dimensional data encoding device may encode the original coordinates using a predetermined fixed bit length. This enables three-dimensional data encoding device to generate a stream with a fixed bit length at a certain time.

[0503]

In the above description, an example has been shown in which the difference, which is the subcoordinates of each point cloud subtracted from the coordinates of each point cloud, is encoded as the non-octree encoding, but is not limited thereto.

[0504]

For example, the three-dimensional data encoding device may sequentially encode a differential value between the coordinates of each point cloud. FIG. 64 is a diagram for describing an operation in this case. For example, in the example shown in FIG. 64, the three-dimensional data encoding device encodes a differential value between coordinates of point cloud PA and predicted coordinates, using the subcoordinates as the predicted coordinates, when encoding point cloud PA. The three-dimensional data encoding device encodes a differential value between point cloud PB and predicted coordinates, using the coordinates of point cloud PA as the predicted coordinates, when encoding point cloud PB. The three-dimensional data encoding device encodes a differential value between point cloud PC and predicted coordinates, using the coordinates of point cloud PB as the predicted coordinates, when encoding point cloud PC. In this manner, the three-dimensional data encoding device may set a scan order to a plurality of point clouds, and encode a differential value between coordinates of a current point cloud to be processed and coordinates of a point cloud immediately before the current point cloud in the scan order.

[0505]

In the above description, the subcoordinates are coordinates in the lower left front corner of the submap, but a location of the subcoordinates is not limited thereto. FIG. 65 to FIG. 67 are diagrams showing other examples of the location of the subcoordinates. The location of the subcoordinates may be set to any coordinates in the current space (submap, space, or volume). In other words, the subcoordinates may be, as stated above, coordinates in the lower left front corner of the current space. As illustrated in FIG. 65, the subcoordinates may be coordinates in a center of the current space. As illustrated in FIG. 66, the subcoordinates may be coordinates in an upper right rear corner of the current space.

The subcoordinates are not limited to being coordinates in the lower left front corner or the upper right rear corner of the current space, but may also be coordinates in any corner of the current space.

[0506]

The location of the subcoordinates may be the same as coordinates of a certain point cloud in the current space (submap, space, or volume). For example, in the example shown in FIG. 67, the coordinates of the subcoordinates coincide with coordinates of point cloud PD.

[0507]

In the present embodiment, an example has been shown that switches between applying octree encoding or non-octree encoding, but is not necessarily limited thereto. For example, the three-dimensional data encoding device may switch between applying a tree structure other than an octree or a non-tree structure other than the tree-structure. For example, the other tree structure is a k-d tree in which splitting is performed using perpendicular planes on one coordinate axis. Note that any other method may be used as the other tree structure.

[0508]

In the present embodiment, an example has been shown in which coordinate information included in a point cloud is encoded, but is not necessarily limited thereto. The three-dimensional data encoding device may encode, for example, color information, a three-dimensional feature quantity, or a feature quantity of visible light using the same method as for the coordinate information. For example, the three-dimensional data encoding device may set an average value of the color information included in each point cloud in the submap to subcolor information, and encode a difference between the color information and the subcolor information of each point cloud.

[0509]

In the present embodiment, an example has been shown in which an encoding

method (octree encoding or non-octree encoding) with good encoding efficiency is selected in accordance with a total number of point clouds and the like, but is not necessarily limited thereto. For example, the three-dimensional data encoding device, which is a server end, may store a bitstream of a point cloud encoded through octree encoding, a bitstream of a point cloud encoded through non-octree encoding, and a bitstream of a point cloud encoded through both methods, and switch the bitstream to be transmitted to the three-dimensional data decoding device, in accordance with a transmission environment or a processing power of the three-dimensional data decoding device.

[0510]

FIG. 68 is a diagram showing an example syntax of a volume when applying octree encoding. The syntax shown in FIG. 68 is basically the same as the syntax shown in FIG. 58, but differs in that each piece of information is information in units of volumes. To be specific, NumOfPoint indicates a total number of point clouds included in the volume. sub_coordinate_x, sub_coordinate_y, and sub_coordinate_z are the subcoordinate information of the volume.

[0511]

diff_x[i], diff_y[i], and diff_z[i] are differential coordinates of an i-th point cloud in the volume. diff_x[i] is a differential value between an x-coordinate of the i-th point cloud and the x-coordinate of the subcoordinates in the volume. diff_y[i] is a differential value between a y-coordinate of the i-th point cloud and the y-coordinate of the subcoordinates in the volume. diff_z[i] is a differential value between a z-coordinate of the i-th point cloud and the z-coordinate of the subcoordinates in the volume.

[0512]

Note that when it is possible to calculate a relative position of the volume in the space, the three-dimensional data encoding device does not need to include the subcoordinate information in a header of the volume. In other words, the three-dimensional data encoding device may calculate the relative position of the volume in the space without including the

subcoordinate information in the header, and use the calculated position as the subcoordinates of each volume.

[0513]

As stated above, the three-dimensional data encoding device according to the present embodiment determines whether to encode, using an octree structure, a current space unit among a plurality of space units (e.g. submaps, spaces, or volumes) included in three-dimensional data (e.g. S1522 in FIG. 59). For example, the three-dimensional data encoding device determines that the current space unit is to be encoded using the octree structure, when a total number of the three-dimensional points included in the current space unit is higher than a predetermined threshold. The three-dimensional data encoding device determines that the current space unit is not to be encoded using the octree structure, when the total number of the three-dimensional points included in the current space unit is lower than or equal to the predetermined threshold.

[0514]

When it is determined that the current space unit is to be encoded using the octree structure (YES in S1522), the three-dimensional data encoding device encodes the current space unit using the octree structure (S1523). When it is determined that the current space unit is not to be encoded using the octree structure (NO in S1522), the three-dimensional data encoding device encodes the current space unit using a different method that is not the octree structure (S1524). For example, in the different method, the three-dimensional data encoding device encodes coordinates of three-dimensional points included in the current space unit. To be specific, in the different method, the three-dimensional data encoding device encodes a difference between reference coordinates of the current space unit and the coordinates of the three-dimensional points included in the current space unit.

[0515]

The three-dimensional data encoding device next appends, to a bitstream, information that indicates whether the current space unit has been encoded using the octree

structure (S1525).

[0516]

This enables the three-dimensional data encoding device to improve encoding efficiency since it is possible to reduce the amount of data of the encoded signal.

[0517]

For example, the three-dimensional data encoding device includes a processor and memory, the processor using the memory to perform the above processes.

[0518]

The three-dimensional data decoding device according to the present embodiment decodes, from a bitstream, information that indicates whether to decode, using an octree structure, a current space unit among a plurality of space units (e.g. submaps, spaces, or volumes) included in three-dimensional data (e.g. S1531 in FIG. 60). When the information indicates that the current space unit is to be decoded using the octree structure (YES in S1532), the three-dimensional data decoding device decodes the current space unit using the octree structure (S1533).

[0519]

When the information indicates not to decode the current space unit using the octree structure (NO in S1532), the three-dimensional data decoding device decodes the current space unit using a different method that is not the octree structure (S1534). For example, in the different method, the three-dimensional data decoding device decodes coordinates of three-dimensional points included in the current space unit. To be specific, in the different method, the three-dimensional data decoding device decodes a difference between reference coordinates of the current space unit and the coordinates of the three-dimensional points included in the current space unit.

[0520]

This enables the three-dimensional data decoding device to improve encoding efficiency since it is possible to reduce the amount of data of the encoded signal.

[0521]

For example, three-dimensional data decoding device includes a processor and memory. The processor uses the memory to perform the above processes.

[0522]

EMBODIMENT 9

In the present embodiment, a method for encoding a tree structure such as an octree structure will be described.

[0523]

It is possible to improve efficiency by identifying an important area and preferentially decoding three-dimensional data of the important area.

[0524]

FIG. 69 is a diagram showing an example of an important area in a three-dimensional map. The important area includes, for example, at least a fixed number of three-dimensional points, among three-dimensional points in the three-dimensional map, having a high feature quantity. The important area may also include, for example, a fixed number of three-dimensional points necessary when, for example, a vehicle-mounted client performs self-location estimation. Alternatively, the important area may also be a face in a three-dimensional model of a person. Such an important area can be defined per application type, and may be switched in accordance therewith.

[0525]

In the present embodiment, occupancy encoding and location encoding are used as a method for representing an octree structure and the like. A bit sequence obtained through occupancy encoding is referred to as occupancy code. A bit sequence obtained through location encoding is referred to as location code.

[0526]

FIG. 70 is a diagram showing an example of an occupancy code. FIG. 70 shows an example of the occupancy code of a quadtree structure. In FIG. 70, occupancy code is

assigned to each node. Each piece of occupancy code indicates whether a three-dimensional point is included in a child node or a leaf of a node. In the case of a quadtree, for example, information, which indicates whether four child nodes or leaves included in each node include three-dimensional points, is expressed with a 4-bit occupancy code. In the case of an octree, information, which indicates whether eight child nodes or leaves included in each node include three-dimensional points, is expressed with an 8-bit occupancy code. Note that an example of a quadtree structure is described here in order to simplify the description, but the same is applicable to an octree structure. As illustrated in FIG. 70, for example, the occupancy code is a bit sequence in which the nodes and leaves have been scanned breadth-first, as described in FIG. 40, etc. In the occupancy code, since a plurality of pieces of three-dimensional point information are decoded in a fixed order, it is not possible to preferentially decode a piece of three-dimensional point information of choice. Note that the occupancy code may also be a bit sequence in which the nodes and leaves have been scanned depth-first, as described in FIG. 40, etc.

[0527]

Hereinafter, location encoding will be described. It is possible to directly decode important portions in the octree structure by using the location code. It is also possible to efficiently encode the important three-dimensional points in deeper levels.

[0528]

FIG. 71 is a diagram for describing location encoding and shows an example of a quadtree structure. In the example shown in FIG. 71, three-dimensional points A–I are represented with a quadtree structure. Three-dimensional points A and C are important three-dimensional points included in the important area.

[0529]

FIG. 72 is a diagram showing occupancy codes and location codes expressing important three-dimensional points A and C in the quadtree structure shown in FIG. 71.

[0530]

In the location encoding, an index of each node present on a path up until a leaf to which a current three-dimensional point belongs that is an encoding target three-dimensional point, and an index of each leaf in the tree structure are encoded. The index here is a numerical value assigned to each node and each leaf. In other words, the index is an identifier for identifying child nodes of a current node. In the case of the quadtree as shown in FIG. 71, indexes between 0 and 3 are shown.

[0531]

In the quadtree structure shown in FIG. 71, for example, leaf A is represented as $0 \rightarrow 2 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 1$ when leaf A is the current three-dimensional point. Since a maximum value of each index in the case of FIG. 71 is 4 (representable as 2-bit value), a bit count necessary for the location code of leaf A is 7×2 bits = 14 bits. Similarly, a bit count necessary when leaf C is the encoding target is 14 bits. Note that in the case of an octree, it is possible to calculate a bit count necessary for 3 bits × leaf depth, since the maximum value of each index is 8 (representable as 3-bit value). Note that the three-dimensional data encoding device may reduce a data amount through entropy encoding after binarizing each index.

[0532]

As illustrated in FIG. 72, in the occupancy code, it is necessary to decode all nodes of upper levels of leaves A and C in order to decode leaves A and C. On the other hand, it is possible to only decode data of leaves A and C in the location code. As illustrated in FIG. 72, this makes it possible to reduce bit count more than with the occupancy code by using the location code.

[0533]

As illustrated in FIG. 72, it is possible to further reduce a code amount by performing dictionary compression such as LZ77 on a portion or all of the location code.

[0534]

An example in which location encoding is applied to three-dimensional points (point

cloud) obtained through LiDAR will be described next. FIG. 73 is a diagram showing the example of the three-dimensional points obtained through LiDAR. The three-dimensional points obtained through LiDAR are sparsely disposed. In other words, when expressing these three-dimensional points with an occupancy code, a number of zero values is high. High three-dimensional precision is required for these three-dimensional points. In other words, the hierarchy of the octree structure becomes deeper.

[0535]

FIG. 74 is a diagram showing an example of such a sparse and deep octree structure. An occupancy code of the octree structure shown in FIG. 74 is a 136-bit value (= 8 bits × 17 nodes). Since the octree structure has a depth of 6 and six three-dimensional points, the location code is 3 bits × 6 × 6 = 108 bits. In other words, the location code is capable of reducing 20% of a code amount of the occupancy code. In this manner, it is possible to reduce the code amount by applying location encoding to the sparse and deep octree structure.

[0536]

Hereinafter, the code amounts of the occupancy code and the location code will be described. When the octree structure has a depth of 10, a maximum number of three-dimensional points is 8^{10} = 1,073,741,824. Bit count L_o of the occupancy code of the octree structure is expressed below.

[0537]

$$L_o = 8 + 8^2 + ... + 8^{10} = 127,133,512 \text{ bits}$$

[0538]

As such, a bit count of one three-dimensional point is 1.143 bits. Note that in the occupancy code, this bit count does not change even if the total number of three-dimensional points included in the octree structure changes.

[0539]

On the other hand, in the location code, a bit count of one three-dimensional point is

directly influenced by the depth of the octree structure. To be specific, a bit count of the location code of one three-dimensional point is 3 bits × depth of 10 = 30 bits.

[0540]

As such, bit count L_l of the location code of the octree structure is expressed below.

[0541]

$$L_l = 30 \times N$$

[0542]

N here is the total number of three-dimensional points included in the octree structure.

[0543]

As such, in the case of $N < L_o / 30 = 40{,}904{,}450.4$, i.e., when the total number of three-dimensional points is lower than 40,904,450, the code amount of the location code is smaller than the code amount of the occupancy code ($L_l < L_o$).

[0544]

In this manner, the code amount of the location code is lower than the code amount of the occupancy code in the case of a low number of three-dimensional points, and the code amount of the location code is higher than the code amount of the occupancy code in the case of a high number of three-dimensional points.

[0545]

As such, the three-dimensional data encoding device may switch between using location encoding or occupancy encoding in accordance with the total number of inputted three-dimensional points. In this case, the three-dimensional data encoding device may append, to header information and the like of the bitstream, information indicating whether the location encoding or the occupancy encoding has been performed.

[0546]

Hereinafter, hybrid encoding, which is a combination of the location encoding and the occupancy encoding, will be described. When encoding a dense important area, hybrid

encoding, which is a combination of the location encoding and the occupancy encoding, is effective. FIG. 75 is a diagram showing this example. In the example shown in FIG. 75, the important three-dimensional points are disposed densely. In this case, the three-dimensional data encoding device performs location encoding on the upper levels at a shallow depth and uses occupancy encoding for the lower levels. To be specific, location encoding is used up until a deepest common node and occupancy encoding is used from the deepest common node up until the deepest level. The deepest common node here is the deepest node among nodes that are the common ancestors of the plurality of important three-dimensional points.

[0547]

Hybrid encoding that prioritizes compression efficiency will be described next. The three-dimensional data encoding device may switch between location encoding or occupancy encoding in accordance with a predetermined rule during encoding of the octree.

[0548]

FIG. 76 is a diagram showing an example of this rule. The three-dimensional data encoding device first checks a percentage of nodes including three-dimensional points at each level (depth). When the percentage is higher than a predetermined threshold value, the three-dimensional data encoding device occupancy encodes several nodes of upper levels of the current level. For example, the three-dimensional data encoding device applies occupancy encoding from the current level to levels up until the deepest common node.

[0549]

For example, in the example of FIG. 76, the percentage of nodes including three-dimensional points in a third level is higher than the predetermined threshold value. As such, the three-dimensional data encoding device applies occupancy encoding from the third level up until the second level including the deepest common node, and applies location encoding to the other levels, i.e., the first level and the fourth level.

[0550]

A method for calculating the above threshold value will be described. One level of the octree structure includes one root node and eight child nodes. As such, in the occupancy encoding, 8 bits are necessary for encoding one level of the octree structure. On the other hand, in the location encoding, 3 bits are necessary per child node including a three-dimensional point. As such, when a total number of nodes including three-dimensional points is higher than 2, occupancy encoding is more effective than location encoding. In other words, in this case, the threshold value is 2.

[0551]

Hereinafter, an example structure of a bitstream generated through the above-mentioned location encoding, occupancy encoding, or hybrid encoding will be described.

[0552]

FIG. 77 is a diagram showing an example of a bitstream generated through location encoding. As illustrated in FIG. 77, the bitstream generated through location encoding includes a header and pieces of location code. Each piece of location code corresponds to one three-dimensional point.

[0553]

This structure enables the three-dimensional data decoding device to individually decode a plurality of three-dimensional points will high precision. Note that FIG. 77 shows an example of a bitstream in the case of a quadtree structure. In the case of an octree structure, each index can take a value between 0 and 7.

[0554]

The three-dimensional data encoding device may entropy encode an index sequence expressing one three-dimensional point after binarizing the index sequence. For example, when the index sequence is 0121, the three-dimensional data encoding device may binarize 0121 into 00011001 and perform arithmetic encoding on this bit sequence.

[0555]

FIG. 78 is a diagram showing an example of a bitstream generated through hybrid encoding when the bitstream includes important three-dimensional points. As illustrated in FIG. 78, location code of upper levels, occupancy code of important three-dimensional points of lower levels, and occupancy code of non-important three-dimensional points of lower levels are disposed in this order. Note that a location code length shown in FIG. 78 expresses a code amount of subsequent location code. An occupancy code amount expresses a code amount of subsequent occupancy code.

[0556]

This structure enables the three-dimensional data decoding device to select a decoding plan in accordance with the type of application.

[0557]

Encoded data of the important three-dimensional points is stored around a head of the bitstream, and encoded data of the non-important three-dimensional points not included in the important area is stored behind the encoded data of the important three-dimensional points.

[0558]

FIG. 79 is a diagram showing a tree structure expressed with the occupancy code of the important three-dimensional points shown in FIG. 78. FIG. 80 is a diagram showing a tree structure expressed with the occupancy code of the non-important three-dimensional points shown in FIG. 78. As illustrated in FIG. 79, information relating to the non-important three-dimensional points is excluded in the occupancy code of the important three-dimensional points. To be specific, since node 0 and node 3 at a depth of 5 do not include important three-dimensional points, value 0 is assigned indicating that node 0 and node 3 do not include three-dimensional points.

[0559]

On the other hand, as illustrated in FIG. 80, information relating to the important three-dimensional points is excluded in the occupancy code of the non-important

three-dimensional points. To be specific, since node 1 at a depth of 5 does not include non-important three-dimensional points, value 0 is assigned indicating that node 1 does not include a three-dimensional point.

[0560]

In this manner, the three-dimensional data encoding device divides the original tree structure into a first tree structure including the important three-dimensional points and a second tree structure including the non-important three-dimensional points, and separately occupancy encodes the first tree structure and the second tree structure. This enables the three-dimensional data decoding device to preferentially decode the important three-dimensional points.

[0561]

An example structure of a bitstream generated through hybrid encoding emphasizing efficiency will be described next. FIG. 81 is a diagram showing the example structure of the bitstream generated through hybrid encoding emphasizing efficiency. As illustrated in FIG. 81, a subtree root location, occupancy code amount, and occupancy code are disposed per subtree in this order. The subtree root location shown in FIG. 81 is the location code of the root of the subtree.

[0562]

In the above structure, the following holds true when only one of location encoding or occupancy encoding is applied to the octree structure.

[0563]

When the length of the location code of the root of the subtree is identical to the depth of the octree structure, the subtree does not include any child nodes. In other words, location encoding is applied to the entire tree structure.

[0564]

When the root of the subtree is identical to the root of the octree structure, occupancy encoding is applied to the entire tree structure.

[0565]

For example, the three-dimensional data decoding device is capable of discerning whether the bitstream includes location code or occupancy code, based on the above rule.

[0566]

The bitstream may include encoding mode information indicating which of location encoding, occupancy encoding, and hybrid encoding is used. FIG. 82 is a diagram showing an example of a bitstream in this case. As illustrated in FIG. 82, for example, 2-bit encoding mode information indicating the encoding mode is appended to the bitstream.

[0567]

(1) "THREE-DIMENSIONAL POINT COUNT" in the location encoding expresses a total number of subsequent three-dimensional points. (2) "OCCUPANCY CODE AMOUNT" in the occupancy encoding expresses a code amount of subsequent occupancy code. (3) "IMPORTANT SUBTREE COUNT" in the hybrid encoding (important three-dimensional points) expresses a total number of subtrees including important three-dimensional points. (4) "OCCUPANCY SUBTREE COUNT" in the hybrid encoding (emphasis on efficiency) expresses a total number of occupancy encoded subtrees.

[0568]

An example syntax used for switching between applying occupancy encoding or location encoding will be described next. FIG. 83 is a diagram showing this example syntax.

[0569]

isleaf shown in FIG. 83 is a flag indicating whether the current node is a leaf. isleaf=1 indicates that the current node is a leaf, and isleaf=0 indicates that the current node is not a leaf.

[0570]

When the current node is a leaf, point_flag is appended to the bitstream. point_flag is a flag indicating whether the current node (leaf) includes a three-dimensional point.

point_flag=1 indicates that the current node includes a three-dimensional point, and point_flag=0 indicates that the current node does not include a three-dimensional point.

[0571]

When the current node is not a leaf, coding_type is appended to the bitstream. coding_type is encoding type information indicating which encoding type has been applied. coding_type=00 indicates that location encoding has been applied, coding_type=01 indicates that occupancy encoding has been applied, and coding_type=10 or 11 indicates that another encoding method has been applied.

[0572]

When the encoding type is location encoding, numPoint, num_idx[i], and idx[i][j] are appended to the bitstream.

[0573]

numPoint indicates a total number of three-dimensional points on which to perform location encoding. num_idx[i] indicates a total number (depth) of indexes from the current node up to three-dimensional point i. When the three-dimensional points on which location encoding is to be performed are all at the same depth, each num_idx[i] has the same value. As such, num_idx may be defined as a common value before the for statement (for (i=0;i<numPoint;i++){) shown in FIG. 83.

[0574]

idx[i][j] indicates a value of a j-th index among indexes from the current node up to three-dimensional point i. In the case of an octree, a bit count of idx[i][j] is 3 bits.

[0575]

Note that, as stated above, the index is an identifier for the identifying child nodes of the current node. In the case of an octree, idx[i][j] indicates a value between 0 and 7. In the case of an octree, there are eight child nodes which respectively correspond to eight subblocks obtained by spatially dividing a current block corresponding to the current node into eight. As such, idx[i][j] may be information indicating a three-dimensional position of

the subblock corresponding to a child node. For example, idx[i][j] may be 3-bit information in total includes three pieces of 1-bit information each indicating a position of each of x, y, and z of the subblock.

[0576]

When the encoding type is occupancy encoding, occupancy_code is appended to the bitstream. occupancy_code is the occupancy code of the current node. In the case of an octree, occupancy_code is an 8-bit bit sequence such as bit sequence "00101000".

[0577]

When a value of an (i+1)-th bit of occupancy_code is 1, processing of the child node begins. In other words, the child node is set as the next current node, and a bit sequence is recursively generated.

[0578]

In the present embodiment, an example is shown in which ends of the octree are expressed by appending leaf information (isleaf, point_flag) to the bitstream, but the present embodiment is not necessarily limited thereto. For example, the three-dimensional data encoding device may append, to a header portion of a start-node (root), a maximum depth from the start-node of the occupancy code up to ends (leaves) including three-dimensional points. The three-dimensional data encoding device may recursively convert information on the child nodes while increasing the depth from the start-node, and may determine as to having arrived at the leaves when the depth becomes the maximum depth. The three-dimensional data encoding device may also append information indicating the maximum depth to the first node where the coding type has become occupancy encoding, and may also append this information to the start-node (root) of the octree.

[0579]

As stated above, the three-dimensional data encoding device may append, to the bitstream, information for switching between occupancy encoding and location encoding as header information of each node.

[0580]

The three-dimensional data encoding device may entropy encode coding_type, numPoint, num_idx, idx, and occupancy_code of each node generated using the above method. For example, the three-dimensional data encoding device arithmetically encodes each value after binarizing each value.

[0581]

In the above syntax, an example is shown of when a depth-first bit sequence of the octree structure is used as the occupancy code, but the present embodiment is not necessarily limited thereto. The three-dimensional data encoding device may use a breadth-first bit sequence of the octree structure as the occupancy code. The three-dimensional data encoding device may append, to the bitstream, information for switching between occupancy encoding and location encoding as header information of each node, also when using a breadth-first bit sequence.

[0582]

In the present embodiment, an example has been shown of an octree structure, but the present embodiment is not necessarily limited thereto, and the above method may be applied to an N-ary (N is an integer of 2 or higher) structure such as a quadtree or a hextree, or another tree structure.

[0583]

Hereinafter, a flow example of an encoding process for switching between applying occupancy encoding or location encoding will be described. FIG. 84 is a flowchart of the encoding process according to the present embodiment.

[0584]

The three-dimensional data encoding device first represents a plurality of three-dimensional points included in three-dimensional data using an octree structure (S1601). The three-dimensional data encoding device next sets a root in the octree structure as a current node (S1602). The three-dimensional data encoding device next generates a bit

sequence of the octree structure by performing a node encoding process on the current node (S1603). The three-dimensional data encoding device next generates a bit sequence by entropy encoding the generated bit sequence (S1604).

[0585]

FIG. 85 is a flowchart of the node encoding process (S1603). The three-dimensional data encoding device first determines whether the current node is a leaf (S1611). When the current node is not a leaf (NO in S1611), the three-dimensional data encoding device sets a leaf flag (isleaf) to 0, and appends the leaf flag to the bit sequence (S1612).

[0586]

The three-dimensional data encoding device next determines whether a total number of child nodes including three-dimensional points is higher than a predetermined threshold value (S1613). Note that three-dimensional data encoding device may append this threshold value to the bit sequence.

[0587]

When the total number of child nodes including three-dimensional points is higher than the predetermined threshold value (YES in S1613), the three-dimensional data encoding device sets the encoding type (coding_type) to occupancy encoding, and appends the encoding type to the bit sequence (S1614).

[0588]

The three-dimensional data encoding device next configures occupancy encoding information, and appends the occupancy encoding information to the bit sequence. To be specific, the three-dimensional data encoding device generates an occupancy code for the current node, and appends the occupancy code to the bit sequence (S1615).

[0589]

The three-dimensional data encoding device next sets the next current node based on the occupancy code (S1616). To be specific, the three-dimensional data encoding device

sets the next current node from an unprocessed child node whose occupancy code is "1".

[0590]

The three-dimensional data encoding device performs the node encoding process on the newly-set current node (S1617). In other words, the process shown in FIG. 85 is performed on the newly-set current node.

[0591]

When all child nodes have not been processed yet (NO in S1618), the processes from step S1616 are performed again. On the other hand, when all of the child nodes have been processed (YES in S1618), the three-dimensional data encoding device ends the node encoding process.

[0592]

In step S1613, when the total number of child nodes including three-dimensional points is lower than or equal to the predetermined threshold value (NO in S1613), the three-dimensional data encoding device sets the encoding type to location encoding, and appends the encoding type to the bit sequence (S1619).

[0593]

The three-dimensional data encoding device next configures location encoding information, and appends the location encoding information to the bit sequence. To be specific, the three-dimensional data encoding device next generates a location code, and appends the location code to the bit sequence (S1620). The location code includes numPoint, num_idx, and idx.

[0594]

In step S1611, when the current node is a leaf (YES in S1611), the three-dimensional data encoding device sets the leaf flag to 1, and appends the leaf flag to the bit sequence (S1621). The three-dimensional data encoding device configures a point flag (point_flag) that is information indicating whether the leaf includes a three-dimensional point, and appends the point flag to the bit sequence (S1622).

[0595]

A flow example of a decoding process for switching between applying occupancy encoding or location encoding will be described next. FIG. 85 is a flowchart of the decoding process according to the present embodiment.

[0596]

The three-dimensional data encoding device generates a bit sequence by entropy decoding the bitstream (S1631). The three-dimensional data decoding device next restores the octree structure by performing a node decoding process on the obtained bit sequence (S1632). The three-dimensional data decoding device next generates the three-dimensional points from the restores octree structure (S1633).

[0597]

FIG. 87 is a flowchart of the node decoding process (S1632). The three-dimensional data decoding device first obtains (decodes) the leaf flag (isleaf) from the bit sequence (S1641). The three-dimensional data decoding device next determines whether the current node is a leaf based on the leaf flag (S1642).

[0598]

When the current node is not a leaf (NO in S1642), the three-dimensional data decoding device obtains the encoding type (coding_type) from the bit sequence (S1643). The three-dimensional data decoding device determines whether the encoding type is occupancy encoding (S1644).

[0599]

When the encoding type is occupancy encoding (YES in S1644), the three-dimensional data decoding device obtains the occupancy encoding information from the bit sequence. To be specific, the three-dimensional data decoding device obtains the occupancy code from the bit sequence (S1645).

[0600]

The three-dimensional data decoding device next sets the next current node based on

the occupancy code (S1646). To be specific, the three-dimensional data decoding device sets the next current node from an unprocessed child node whose occupancy code is "1".

[0601]

The three-dimensional data decoding device next performs the node decoding process on the newly-set current node (S1647). In other words, the process shown in FIG. 87 is performed on the newly-set current node.

[0602]

When all child nodes have not been processed yet (NO in S1648), the processes from step S1646 are performed again. On the other hand, when all of the child nodes have been processed (YES in S1648), the three-dimensional data decoding device ends the node decoding process.

[0603]

In step 1644, when the encoding type is location encoding (NO in S1644), the three-dimensional data decoding device obtains the location encoding information from the bit sequence. To be specific, the three-dimensional data decoding device obtains the location code from the bit sequence (S1649). The location code includes numPoint, num_idx, and idx.

[0604]

In step S1642, when the current node is a leaf (YES in S1642), the three-dimensional data decoding device obtains, from the bit sequence, the point flag (point_flag) that is the information indicating whether the leaf includes a three-dimensional point (S1650).

[0605]

Note that in the present embodiment, an example has been shown in which the encoding type is switched per node, but the present embodiment is not necessarily limited thereto. The encoding type may be fixed per volume, space, or world unit. In this case, the three-dimensional data encoding device may append encoding type information to header

information of the volume, space, or world.

[0606]

As stated above, the three-dimensional data encoding device according to the present embodiment: generates first information in which an N-ary (N is an integer of 2 or higher) tree structure of a plurality of three-dimensional points included in three-dimensional data is expressed using a first formula (location encoding); and generates a bitstream including the first information. The first information includes pieces of three-dimensional point information (location code) each associated with a corresponding one of the plurality of three-dimensional points. The pieces of three-dimensional point information each include indexes (idx) each associated with a corresponding one of a plurality of levels in the N-ary tree structure. The indexes each indicate a subblock, among N subblocks belonging to a corresponding one of the plurality of levels, to which a corresponding one of the plurality of three-dimensional points belongs.

[0607]

In other words, the pieces of three-dimensional point information each indicate a path until the corresponding one of the plurality of three-dimensional points in the N-ary tree structure. The indexes each indicate a child node, among N child nodes belonging to a corresponding layer (node), included on the path.

[0608]

This enables the three-dimensional data encoding method to generate a bitstream from which the three-dimensional points can be selectively decoded.

[0609]

For example, the pieces of three-dimensional point information (location code) each include information (num_idx) indicating a total number of the indexes included in the piece of three-dimensional point information. In other words, the information indicates a depth (layer count) until a corresponding three-dimensional point in the N-ary tree structure.

[0610]

For example, the first information includes information (numPoint) indicating a total number of the pieces of three-dimensional point information included in the first information. In other words, the information indicates a total number of three-dimensional points included in the N-ary tree structure.

[0611]

For example, N is 8, and the indexes are each a 3-bit value.

[0612]

For example, in the three-dimensional data encoding device, a first encoding mode is used for generating the first information, and a second encoding mode is used for (i) generating second information (occupancy code) in which the N-ary tree structure is expressed using a second formula (occupancy encoding) and (ii) generating a bitstream including the second information. The second information includes pieces of 1-bit information each of which (i) is associated with a corresponding one of a plurality of subblocks belonging to the plurality of levels in the N-ary tree structure and (ii) indicates whether a three-dimensional point is present in the corresponding one of the plurality of subblocks.

[0613]

For example, the three-dimensional data encoding device uses the first encoding mode when a total number of the plurality of three-dimensional points is lower than or equal to a predetermined threshold value, and the second encoding mode may be used when the total number of the plurality of three-dimensional points is higher than the predetermined threshold value. This enables the three-dimensional data encoding method to reduce a code amount of the bitstream.

[0614]

For example, the first information and the second information each include information (encoding mode information) indicating whether the N-ary tree structure is expressed using the first formula or the second formula.

[0615]

For example, as illustrated in FIG. 75 and the like, the three-dimensional data encoding device uses the first encoding mode for one portion of the N-ary tree structure and the second encoding mode for another portion of the N-ary tree structure.

[0616]

For example, the three-dimensional data encoding device includes a processor and memory, the processor using the memory to perform the above processes.

[0617]

The three-dimensional data decoding device according to the present embodiment obtains, from a bitstream, first information (location code) in which an N-ary (N is an integer of 2 or higher) tree structure of a plurality of three-dimensional points included in three-dimensional data is expressed using a first formula (location encoding). The first information includes pieces of three-dimensional point information (location code) each associated with a corresponding one of the plurality of three-dimensional points. The pieces of three-dimensional point information each include indexes (idx) each associated with a corresponding one of a plurality of levels in the N-ary tree structure. The indexes each indicate a subblock, among N subblocks belonging to a corresponding one of the plurality of levels, to which a corresponding one of the plurality of three-dimensional points belongs.

[0618]

In other words, the pieces of three-dimensional point information each indicate a path until the corresponding one of the plurality of three-dimensional points in the N-ary tree structure. The indexes each indicate a child node, among N child nodes belonging to a corresponding layer (node), included on the path.

[0619]

The three-dimensional data decoding method further restores, using a corresponding one of the pieces of three-dimensional point information, a three-dimensional point associated with the corresponding one of the pieces of three-dimensional point information.

[0620]

This enables the three-dimensional data decoding device to selectively generate the three-dimensional points from the bitstream.

[0621]

For example, the pieces of three-dimensional point information (location code) each include information (num_idx) indicating a total number of the indexes included in the piece of three-dimensional point information. In other words, the information indicates a depth (layer count) until a corresponding three-dimensional point in the N-ary tree structure.

[0622]

For example, the first information includes information (numPoint) indicating a total number of the pieces of three-dimensional point information included in the first information. In other words, the information indicates a total number of three-dimensional points included in the N-ary tree structure.

[0623]

For example, N is 8, and the indexes are each a 3-bit value.

[0624]

For example, the three-dimensional data decoding device further obtains, from a bitstream, second information (occupancy code) in which an N-ary tree structure is expressed using a second formula (occupancy encoding). The three-dimensional data decoding device restores the plurality of three-dimensional points using the second information. The second information includes pieces of 1-bit information each of which (i) is associated with a corresponding one of a plurality of subblocks belonging to the plurality of levels in the N-ary tree structure and (ii) indicates whether a three-dimensional point is present in the corresponding one of the plurality of subblocks.

[0625]

For example, the first information and the second information each include information (encoding mode information) indicating whether the N-ary tree structure is

expressed using the first formula or the second formula.

[0626]

For example, as illustrated in FIG. 75 and the like, one portion of the N-ary tree structure is expressed using the first formula and another portion of the N-ary tree structure is expressed using the second formula.

[0627]

For example, the three-dimensional data decoding device includes a processor and memory, the processor using the memory to perform the above processes.

[0628]

EMBODIMENT 10

In the present embodiment, another example of the method of encoding a tree structure such as an octree structure will be described.FIG. 88 is a diagram illustrating an example of a tree structure according to the present embodiment. Specifically, FIG. 88 shows an example of a quadtree structure.

[0629]

A leaf including a three-dimensional point is referred to as a valid leaf, and a leaf including no three-dimensional point is referred to as an invalid leaf. A branch having the number of valid leaves greater than or equal to a threshold value is referred to as a dense branch. A branch having the number of valid leaves less than the threshold value is referred to as a sparse branch.

[0630]

A three-dimensional data encoding device calculates the number of three-dimensional points (i.e., the number of valid leaves) included in each branch in a layer of a tree structure. FIG. 88 shows an example in which a threshold value is 5. In this example, two branches are present in layer 1. Since the left branch includes seven three-dimensional points, the left branch is determined as a dense branch. Since the right branch includes two three-dimensional points, the right branch is determined as a sparse

branch.

[0631]

FIG. 89 is a graph showing an example of the number of valid leaves (3D points) of each branch in layer 5. The horizontal axis of FIG. 89 indicates an index that is an identification number of the branch in layer 5. As clearly shown in FIG. 89, specific branches include many three-dimensional points, compared to other branches. Occupancy encoding is more effective for such dense branches than for sparse branches.

[0632]

The following describes how occupancy encoding and location encoding are applied. FIG. 90 is a diagram illustrating a relationship between encoding schemes to be applied and the number of three-dimensional points (the number of valid leaves) included in each branch in layer 5. As illustrated in FIG. 90, the three-dimensional data encoding device applies the occupancy encoding to dense branches, and applies the location encoding to sparse branches. As a result, it is possible to improve the coding efficiency.

[0633]

FIG. 91 is a diagram illustrating an example of a dense branch area in LiDAR data. As illustrated in FIG. 91, a three-dimensional point density calculated from the number of three-dimensional points included in each branch varies from area to area.

[0634]

Separating dense three-dimensional points (branch) and sparse three-dimensional points (branch) brings the following advantage. A three-dimensional point density is higher with a decreasing distance to a LiDAR sensor. Consequently, separating branches in accordance with sparseness and denseness enables division in a distance direction. Such division is effective for specific applications. Using a method other than the occupancy encoding is effective for sparse branches.

[0635]

In the present embodiment, the three-dimensional data encoding device separates an

inputted three-dimensional point cloud into two or more three-dimensional point sub-clouds, and applies a different encoding method to each of the two or more three-dimensional point sub-clouds.

[0636]

For example, the three-dimensional data encoding device separates an inputted three-dimensional point cloud into three-dimensional point sub-cloud A (dense three-dimensional point cloud: dense cloud) including a dense branch, and three-dimensional point sub-cloud B (sparse three-dimensional point cloud: sparse cloud). FIG. 92 is a diagram illustrating an example of three-dimensional point sub-cloud A (dense three-dimensional point cloud) including a dense branch which is separated from the tree structure illustrated in FIG. 88. FIG. 93 is a diagram illustrating an example of three-dimensional point sub-cloud B (sparse three-dimensional point cloud) including a sparse branch which is separated from the tree structure illustrated in FIG. 88.

[0637]

Next, the three-dimensional data encoding device encodes three-dimensional point sub-cloud A using the occupancy encoding, and encodes three-dimensional point sub-cloud B using the location encoding.

[0638]

It should be noted that although the example has been described above in which different encoding schemes (the occupancy encoding and the location encoding) are applied as different encoding methods, for example, the three-dimensional data encoding device may apply the same encoding scheme to three-dimensional point sub-cloud A and three-dimensional point sub-cloud B, and may use different parameters in encoding three-dimensional point sub-cloud A and three-dimensional point sub-cloud B.

[0639]

The following describes a procedure for a three-dimensional data encoding process performed by the three-dimensional data encoding device. FIG. 94 is a flowchart of a

three-dimensional data encoding process performed by the three-dimensional data encoding device according to the present embodiment.

[0640]

First, the three-dimensional data encoding device separates an inputted three-dimensional point cloud into three-dimensional point sub-clouds (S1701). The three-dimensional data encoding device may perform this separation automatically or based on information inputted by a user. For example, the user may specify a range of three-dimensional point sub-clouds. As for an example of automatic separation, for example, when input data is LiDAR data, the three-dimensional data encoding device performs the separation using distance information indicating a distance to each point cloud. Specifically, the three-dimensional data encoding device separates point clouds within a certain range from a measurement point, and point clouds outside the certain range. In addition, the three-dimensional data encoding device may perform the separation using information indicating, for example, important areas and unimportant areas.

[0641]

Next, the three-dimensional data encoding device generates encoded data (encoded bitstream) by encoding three-dimensional point sub-cloud A using method A (S1702). Besides, the three-dimensional data encoding device generates encoded data by encoding three-dimensional point sub-cloud B using method B (S1703). It should be noted that the three-dimensional data encoding device may encode three-dimensional point sub-cloud B using method A. In this case, the three-dimensional data encoding device encodes three-dimensional point sub-cloud B using a parameter different from an encoding parameter used in encoding three-dimensional point sub-cloud A. For example, this parameter may be a quantization parameter. For example, the three-dimensional data encoding device encodes three-dimensional point sub-cloud B using a quantization parameter greater than a quantization parameter used in encoding three-dimensional point sub-cloud A. In this case, the three-dimensional data encoding device may append information indicating a quantization

parameter used in encoding each of three-dimensional point sub-clouds, to a header of encoded data of the three-dimensional point sub-cloud.

[0642]

Then, the three-dimensional data encoding device generates a bitstream by combining the encoded data obtained in step S1702 and the encoded data obtained in step S1703 (S1704).

[0643]

Moreover, the three-dimensional data encoding device may encode, as header information of the bitstream, information for decoding each three-dimensional point sub-cloud. For example, the three-dimensional data encoding device may encode information as described below.

[0644]

The header information may include information indicating the number of encoded three-dimensional sub-points. In this example, this information indicates 2.

[0645]

The header information may include information indicating the number of three-dimensional points included in each three-dimensional point sub-cloud, and encoding methods. In this example, this information indicates the number of three-dimensional points included in three-dimensional point sub-cloud A, the encoding method (method A) applied to three-dimensional point sub-cloud A, the number of three-dimensional points included in three-dimensional point sub-cloud B, and the encoding method (method B) applied to three-dimensional point sub-cloud B.

[0646]

The header information may include information for identifying the start position or end position of encoded data of each three-dimensional point sub-cloud.

[0647]

Moreover, the three-dimensional data encoding device may encode

three-dimensional point sub-cloud A and three-dimensional point sub-cloud B in parallel. Alternatively, the three-dimensional data encoding device may encode three-dimensional point sub-cloud A and three-dimensional point sub-cloud B in sequence.

[0648]

A method of separation into three-dimensional point sub-clouds is not limited to the above method. For example, the three-dimensional data encoding device changes a separation method, performs encoding using each of separation methods, and calculates the coding efficiency of encoded data obtained using each separation method. Subsequently, the three-dimensional data encoding device selects a separation method having the highest coding efficiency from the separation methods. For example, the three-dimensional data encoding device may (i) separate three-dimensional point clouds in each of layers, (ii) calculate coding efficiency in each of the cases, (iii) select a separation method (i.e., a layer in which separation is performed) having the highest coding efficiency from separation methods, (iv) generate three-dimensional point sub-clouds using the selected separation method, and (v) perform encoding.

[0649]

Moreover, when combining encoded data, the three-dimensional data encoding device may place encoding information of a more important three-dimensional point sub-cloud in a position closer to the head of a bitstream. Since this enables a three-dimensional data decoding device to obtain important information by only decoding the head of the bitstream, the three-dimensional data decoding device can obtain the important information quickly.

[0650]

The following describes a procedure for a three-dimensional data decoding process performed by the three-dimensional data decoding device. FIG. 95 is a flowchart of a three-dimensional data decoding process performed by the three-dimensional data decoding device according to the present embodiment.

[0651]

First, for example, the three-dimensional data decoding device obtains a bitstream generated by the above three-dimensional data encoding device. Next, the three-dimensional data decoding device separates, from the obtained bitstream, encoded data of three-dimensional point sub-cloud A and encoded data of three-dimensional point sub-cloud B (S1711). Specifically, the three-dimensional data decoding device decodes, from header information of the bitstream, information for decoding each three-dimensional point sub-cloud, and separates encoded data of each three-dimensional point sub-cloud using the information.

[0652]

Then, the three-dimensional data decoding device obtains three-dimensional point sub-cloud A by decoding the encoded data of three-dimensional point sub-cloud A using method A (S1712). In addition, the three-dimensional data decoding device obtains three-dimensional point sub-cloud B by decoding the encoded data of three-dimensional point sub-cloud B using method B (S1713). After that, the three-dimensional data decoding device combines three-dimensional point sub-cloud A and three-dimensional point sub-cloud B (S1714).

[0653]

It should be noted that the three-dimensional data decoding device may decode three-dimensional point sub-cloud A and three-dimensional point sub-cloud B in parallel. Alternatively, the three-dimensional data decoding device may decode three-dimensional point sub-cloud A and three-dimensional point sub-cloud B in sequence.

[0654]

Moreover, the three-dimensional data decoding device may decode a necessary three-dimensional point sub-cloud. For example, the three-dimensional data decoding device may decode three-dimensional point sub-cloud A and need not decode three-dimensional point sub-cloud B. For example, when three-dimensional point sub-cloud

A is a three-dimensional point cloud included in an important area of LiDAR data, the three-dimensional data decoding device decodes the three-dimensional point cloud included in the important area. Self-location estimation etc. in a vehicle or the like is performed using the three-dimensional point cloud included in the important area.

[0655]

The following describes a specific example of an encoding process according to the present embodiment. FIG. 96 is a flowchart of a three-dimensional data encoding process performed by the three-dimensional data encoding device according to the present embodiment.

[0656]

First, the three-dimensional data encoding device separates inputted three-dimensional points into a sparse three-dimensional point cloud and a dense three-dimensional point cloud (S1721). Specifically, the three-dimensional data encoding device counts the number of valid leaves of a branch in a layer of an octree structure. The three-dimensional data encoding device sets each branch as a dense branch or a sparse branch in accordance with the number of valid leaves of the branch. Subsequently, the three-dimensional data encoding device generates a three-dimensional point sub-cloud (a dense three-dimensional point cloud) obtained by gathering dense branches, and a three-dimensional point sub-cloud (a sparse three-dimensional point cloud) obtained by gathering sparse branches.

[0657]

Next, the three-dimensional data encoding device generates encoded data by encoding the sparse three-dimensional point cloud (S1722). For example, the three-dimensional data encoding device encodes a sparse three-dimensional point cloud using the location encoding.

[0658]

Furthermore, the three-dimensional data encoding device generates encoded data by

encoding the dense three-dimensional point cloud (S1723). For example, the three-dimensional data encoding device encodes a dense three-dimensional point cloud using the occupancy encoding.

[0659]

Then, the three-dimensional data encoding device generates a bitstream by combining the encoded data of the sparse three-dimensional point cloud obtained in step S1722 and the encoded data of the dense three-dimensional point cloud obtained in step S1723 (S1724).

[0660]

Moreover, the three-dimensional data encoding device may encode, as header information of the bitstream, information for decoding the sparse three-dimensional point cloud and the dense three-dimensional point cloud. For example, the three-dimensional data encoding device may encode information as described below.

[0661]

The header information may include information indicating the number of encoded three-dimensional point sub-clouds. In this example, this information indicates 2.

[0662]

The header information may include information indicating the number of three-dimensional points included in each three-dimensional point sub-cloud, and encoding methods. In this example, this information indicates the number of three-dimensional points included in the sparse three-dimensional point cloud, the encoding method (location encoding) applied to the sparse three-dimensional point cloud, the number of three-dimensional points included in the dense three-dimensional point cloud, and the encoding method (occupancy encoding) applied to the dense three-dimensional point cloud.

[0663]

The header information may include information for identifying the start position or end position of encoded data of each three-dimensional point sub-cloud. In this example,

this information indicates at least one of the start position and end position of the encoded data of the sparse three-dimensional point cloud or the start position and end position of the encoded data of the dense three-dimensional point cloud.

[0664]

Moreover, the three-dimensional data encoding device may encode the sparse three-dimensional point cloud and the dense three-dimensional point cloud in parallel. Alternatively, the three-dimensional data encoding device may encode the sparse three-dimensional point cloud and the dense three-dimensional point cloud in sequence.

[0665]

The following describes a specific example of a three-dimensional data decoding process. FIG. 97 is a flowchart of a three-dimensional data decoding process performed by the three-dimensional data decoding device according to the present embodiment.

[0666]

First, for example, the three-dimensional data decoding device obtains a bitstream generated by the above three-dimensional data encoding device. Next, the three-dimensional data decoding device separates, from the obtained bitstream, encoded data of a sparse three-dimensional point cloud and encoded data of a dense three-dimensional point cloud (S1731). Specifically, the three-dimensional data decoding device decodes, from header information of the bitstream, information for decoding each three-dimensional point sub-cloud, and separates encoded data of each three-dimensional point sub-cloud using the information. In this example, the three-dimensional data decoding device separates, from the bitstream, the encoded data of the sparse three-dimensional point cloud and the encoded data of the dense three-dimensional point cloud using the header information.

[0667]

Then, the three-dimensional data decoding device obtains the sparse three-dimensional point cloud by decoding the encoded data of the sparse three-dimensional point cloud (S1732). For example, the three-dimensional data decoding device decodes the

sparse three-dimensional point cloud using location decoding for decoding encoded data obtained as a result of the location encoding.

[0668]

In addition, the three-dimensional data decoding device obtains the dense three-dimensional point cloud by decoding the encoded data of the dense three-dimensional point cloud (S1733). For example, the three-dimensional data decoding device decodes the dense three-dimensional point cloud using occupancy decoding for decoding encoded data obtained as a result of the occupancy encoding.

[0669]

After that, the three-dimensional data decoding device combines the sparse three-dimensional point cloud obtained in step S1732 and the dense three-dimensional point cloud obtained in step S1733 (S1734).

[0670]

It should be noted that the three-dimensional data decoding device may decode the sparse three-dimensional point cloud and the dense three-dimensional point cloud in parallel. Alternatively, the three-dimensional data decoding device may decode the sparse three-dimensional point cloud and the dense three-dimensional point cloud in sequence.

[0671]

Moreover, the three-dimensional data decoding device may decode part of necessary three-dimensional point sub-clouds. For example, the three-dimensional data decoding device may decode a dense three-dimensional point cloud and need not decode a sparse three-dimensional point cloud. For example, when a dense three-dimensional point cloud is a three-dimensional point cloud included in an important area of LiDAR data, the three-dimensional data decoding device decodes the three-dimensional point cloud included in the important area. Self-location estimation etc. in a vehicle or the like is performed using the three-dimensional point cloud included in the important area.

[0672]

FIG. 98 is a flowchart of an encoding process according to the present embodiment. First, the three-dimensional data encoding separates an inputted three-dimensional point cloud into a sparse three-dimensional point cloud and a dense three-dimensional point cloud (S1741).

[0673]

Next, the three-dimensional data encoding device generates encoded data by encoding the dense three-dimensional point cloud (S1742). Then, the three-dimensional data encoding device generates encoded data by encoding the sparse three-dimensional point cloud (S1743). Finally, the three-dimensional data encoding device generates a bitstream by combining the encoded data of the sparse three-dimensional point cloud obtained in step S1742 and the encoded data of the dense three-dimensional point cloud obtained in step S1743 (S1744).

[0674]

FIG. 99 is a flowchart of a decoding process according to the present embodiment. First, the three-dimensional data decoding device extracts, from a bitstream, encoded data of a sparse three-dimensional point cloud and encoded data of a dense three-dimensional (S1751). Next, the three-dimensional data decoding device obtains decoded data of the dense three-dimensional point cloud by decoding the encoded data of the dense three-dimensional point cloud (S1752). Then, the three-dimensional data decoding device obtains decoded data of the sparse three-dimensional point cloud by decoding the encoded data of the sparse three-dimensional point cloud (S1753). Finally, the three-dimensional data decoding device generates a three-dimensional point cloud by combining the decoded data of the dense three-dimensional point cloud obtained in step S1752 and the decoded data of the sparse three-dimensional point cloud obtained in step S1753 (S1754).

[0675]

It should be noted that the three-dimensional data encoding device and the three-dimensional data decoding device may encode and decode any one of a dense

three-dimensional point cloud and a sparse three-dimensional point cloud first. In addition, encoding processes or decoding processes may be performed in parallel using processors etc.

[0676]

Moreover, the three-dimensional data encoding device may encode one of a dense three-dimensional point cloud and a sparse three-dimensional point cloud. For example, when a dense three-dimensional point cloud includes important information, the three-dimensional data encoding device extracts the dense three-dimensional point cloud and a sparse three-dimensional point cloud from an inputted three-dimensional point cloud, and encode the dense three-dimensional point cloud but does not encode the sparse three-dimensional point cloud. This enables the three-dimensional data encoding device to append the important information to a stream while reducing an amount of bit. For example, when, between a server and a client, the client sends to the server a transmission request for three-dimensional point cloud information about the surroundings of the client, the server encodes important information about the surroundings of the client as a dense three-dimensional point cloud and transmits the encoded important information to the client. This enables the server to transmit the information requested by the client while reducing a network bandwidth.

[0677]

Moreover, the three-dimensional data decoding device may decode one of a dense three-dimensional point cloud and a sparse three-dimensional point cloud. For example, when a dense three-dimensional point cloud includes important information, the three-dimensional data decoding device decodes the dense three-dimensional point cloud but does not decode a sparse three-dimensional point cloud. This enables the three-dimensional data decoding device to obtain necessary information while reducing a processing load of the decoding process.

[0678]

FIG. 100 is a flowchart of the process of separating three-dimensional points

(S1741) illustrated in FIG. 98. First, the three-dimensional data encoding device sets layer L and threshold value TH (S1761). It should be noted that the three-dimensional data encoding device may append information indicating set layer L and threshold value TH, to a bitstream. In other words, the three-dimensional data encoding device may generate a bitstream including information indicating set layer L and threshold value TH.

[0679]

Next, the three-dimensional data encoding device moves a target position from a root of an octree to a lead branch in layer L. In other words, the three-dimensional data encoding device selects the lead branch in layer L as a current branch (S1762).

[0680]

Then, the three-dimensional data encoding device counts the number of valid leaves of the current branch in layer L (S1763). When the number of the valid leaves of the current branch is greater than threshold value TH (YES in S1764), the three-dimensional data encoding device registers the current branch as a dense branch with a dense three-dimensional point cloud (S1765). In contrast, when the number of the valid leaves of the current branch is less than threshold value TH (NO in S1764), the three-dimensional data encoding device registers the current branch as a sparse branch with a sparse three-dimensional point cloud (S1766).

[0681]

When processing of all branches in layer L is not completed (NO in S1767), the three-dimensional data encoding device moves the target position to the next branch in layer L. In other words, the three-dimensional data encoding device selects the next branch in layer L as a current branch (S1768). And then, the three-dimensional data encoding device performs step S1763 and the subsequent steps on the selected next current branch.

[0682]

The above-described process is repeated until the processing of all the branches in layer L is completed (YES in S1767).

[0683]

It should be noted that although layer L and threshold value TH are preset in the above description, the present embodiment is not necessarily limited to this. For example, the three-dimensional data encoding device sets different combinations of layer L and threshold value TH, generates a dense three-dimensional point cloud and a sparse three-dimensional point cloud using each of the combinations, and encodes the dense three-dimensional point cloud and the sparse three-dimensional point cloud. The three-dimensional data encoding device finally encodes the dense three-dimensional point cloud and the sparse three-dimensional point cloud using, among the combinations, a combination of layer L and threshold value TH having the highest coding efficiency for encoded data generated. This makes it possible to improve the coding efficiency. Moreover, for example, the three-dimensional data encoding device may calculate layer L and threshold value TH. For example, the three-dimensional data encoding device may set, to layer L, a value half as much as the maximum value of layers included in a tree structure. Furthermore, the three-dimensional data encoding device may set, to threshold value TH, a value half as much as a total number of three-dimensional points included in the tree structure.

[0684]

In the above description, the example has been shown in which the inputted three-dimensional point cloud is separated into two types of three-dimensional point cloud, that is, the dense three-dimensional point cloud and the sparse three-dimensional point cloud. The three-dimensional data encoding device, however, may separate the inputted three-dimensional point cloud into at least three types of three-dimensional point cloud. For example, when the number of valid leaves of a current branch is greater than or equal to first threshold value TH1, the three-dimensional data encoding device classifies the current branch into a first dense three-dimensional point cloud, and when the number of the valid leaves of the current branch is less than first threshold value TH1 and greater than or equal to second

threshold value TH2, the three-dimensional data encoding device classifies the current branch into a second dense three-dimensional point cloud. When the number of the valid leaves of the current branch is less than second threshold value TH2 and greater than or equal to third threshold value TH3, the three-dimensional data encoding device classifies the current branch into a first sparse three-dimensional point cloud, and when the number of the valid leaves of the current branch is less than third threshold value TH3, the three-dimensional data encoding device classifies the current branch into a second sparse three-dimensional point cloud.

[0685]

The following describes an example of a syntax of encoded data of a three-dimensional point cloud according to the present embodiment. FIG. 101 is a diagram illustrating an example of this syntax. pc_header() is, for example, header information of inputted three-dimensional points.

[0686]

num_sub_pc illustrated in FIG. 101 indicates the number of three-dimensional point sub-clouds. numPoint[i] indicates the number of three-dimensional points included in the i-th three-dimensional point sub-cloud. coding_type[i] is coding type information indicating a coding type (an encoding scheme) applied to the i-th three-dimensional point sub-cloud. For example, coding_type=00 indicates that the location encoding has been applied. coding_type=01 indicates that the occupancy encoding has been applied. coding_type=10 or 11 indicates that another encoding scheme has been applied.

[0687]

data_sub_cloud() is encoded data of the i-th three-dimensional point sub-cloud. coding_type_00_data is encoded data to which a coding type of 00 indicated by coding_type has been applied, and is encoded data to which the location encoding has been applied, for example. coding_type_01_data is encoded data to which a coding type of 01 indicated by coding_type has been applied, and is encoded data to which the occupancy encoding has been applied, for example.

[0688]

end_of_data is end information indicating the end of encoded data. For example, a constant bit sequence not used for encoded data is assigned to end_of_data. This enables the three-dimensional data decoding device to, for example, skip decoding of encoded data that need not be decoded, by searching a bitstream for a bit sequence of end_of_data.

[0689]

It should be noted that the three-dimensional data encoding device may entropy encode the encoded data generated by the above-described method. For example, the three-dimensional data encoding device binarizes each value and performs arithmetic encoding on the binarized value.

[0690]

Although the example of the quadtree structure or the octree structure has been shown in the present embodiment, the present embodiment is not necessarily limited to this. The above-described method may be applied to an N-ary (N is an integer greater than or equal to 2) tree, such as a binary tree and a hexadecatree, or another tree structure.

[0691]

VARIATION

In the above example, as illustrated in FIG. 92 and FIG. 93, the tree structure is encoded that includes the dense branch and the upper layer (the tree structure from the root of the whole tree structure to the root of the dense branch), and the tree structure is encoded that includes the sparse branch and the upper layer (the tree structure from the root of the whole tree structure to the root of the sparse branch). In the present variation, the three-dimensional data encoding device separates a dense branch and a sparse branch, and encodes the dense branch and the sparse branch. In other words, a tree structure to be encoded does not include a tree structure of an upper layer. For example, the three-dimensional data encoding device applies the occupancy encoding to a dense branch, and applies the location encoding to a sparse branch.

[0692]

FIG. 102 is a diagram illustrating an example of a dense branch separated from the tree structure illustrated in FIG. 88. FIG. 103 is a diagram illustrating an example of a sparse branch separated from the tree structure illustrated in FIG. 88. In the present variation, the tree structures illustrated in FIG.102 and FIG. 103 are encoded.

[0693]

The three-dimensional data encoding device encodes information indicating a position of a branch instead of encoding a tree structure of an upper layer. For example, this information indicates a position of a root of a branch.

[0694]

For example, the three-dimensional data encoding device encodes, as encoded data of a dense branch, layer information indicating a layer in which the dense branch is generated, and branch information indicating what number branch in the layer the dense branch is. This enables the three-dimensional data decoding device to decode the layer information and the branch information from a bitstream, and grasp which three-dimensional point cloud of what number branch in which layer the decoded dense branch is. Likewise, the three-dimensional data encoding device encodes, as encoded data of a sparse branch, layer information indicating a layer in which the sparse branch is generated, and branch information indicating what number branch in the layer the sparse branch is present, using these layer information and branch information.

[0695]

This enables the three-dimensional data decoding device to decode the layer information and the branch information from a bitstream, and grasp which three-dimensional point cloud of what number branch in which layer the decoded sparse branch is, using these layer information and branch information. Accordingly, since it is possible to reduce overhead resulting from encoding information of a layer higher than the dense branch or the sparse branch, it is possible to improve the coding efficiency.

[0696]

It should be noted that branch information may indicate a value assigned to each branch in a layer indicated by layer information. Moreover, branch information may indicate a value assigned to each node from a root of an octree as a starting point. In this case, layer information need not be encoded. Furthermore, the three-dimensional data encoding device may generate dense branches and sparse branches.

[0697]

FIG. 104 is a flowchart of an encoding process according to the present variation. First, the three-dimensional data encoding device generates one or more sparse branches and one or more dense branches from an inputted three-dimensional point cloud (S1771).

[0698]

Next, the three-dimensional data encoding device generates encoded data by encoding each of the one or more dense branches (S1772). Then, the three-dimensional data encoding device determines whether encoding of all the dense branches generated in step S1771 is completed (S1774).

[0699]

When the encoding of all the dense branches is not completed (NO in S1773), the three-dimensional data encoding device selects the next dense branch (S1774) and generates encoded data by encoding the selected dense branch (S1772).

[0700]

On the other hand, when the encoding of all the dense branches is completed (YES in S1773), the three-dimensional data encoding device generates encoded data by encoding each of the one or more sparse branches (S1775). Next, the three-dimensional data encoding device determines whether encoding of all the sparse branches generated in step S1771 is completed (S1776).

[0701]

When the encoding of all the sparse branches is not completed (NO in S1776), the

three-dimensional data encoding device selects the next sparse branch (S1777) and generates encoded data by encoding the selected sparse branch (S1775).

[0702]

On the other hand, when the encoding of all the sparse branches is completed (YES in S1776), the three-dimensional data encoding device combines the encoded data generated in steps S1772 and S1775 to generate a bitstream (S1778).

[0703]

FIG. 105 is a flowchart of a decoding process according to the present variation. First, the three-dimensional data decoding device extracts one or more encoded data items of respective dense branches, and one or more encoded data items of respective sparse branches, from a bitstream (S1781). Next, the three-dimensional data decoding device obtains decoded data of each of the dense branches by decoding the encoded data of the dense branch (S1782).

[0704]

Then, the three-dimensional data decoding device determines whether decoding of the encoded data items of all the dense branches extracted in step S1781 is completed (S1783). When the decoding of the encoded data items of all the dense branches is not completed (NO in S1783), the three-dimensional data decoding device selects the encoded data of the next dense branch (S1784) and obtains decoded data of the dense branch by decoding the selected encoded data of the dense branch (S1782).

[0705]

On the other hand, when the decoding of the encoded data items of all the dense branches is completed (YES in S1783), the three-dimensional data decoding device obtains decoded data of each of the sparse branches by decoding the encoded data of the sparse branch (S1785).

[0706]

After that, the three-dimensional data decoding device determines whether decoding

of the encoded data items of all the sparse branches extracted in step S1781 is completed (S1786). When the decoding of the encoded data items of all the sparse branches is not completed (NO in S1786), the three-dimensional data decoding device selects the encoded data of the next sparse branch (S1787) and obtains decoded data of the sparse branch by decoding the selected encoded data of the sparse branch (S1785).

[0707]

On the other hand, when the decoding of the encoded data items of all the sparse branches is completed (YES in S1786), the three-dimensional data decoding device combines the decoded data obtained in steps S1782 and S1785 to generate a three-dimensional point cloud (S1788).

[0708]

It should be noted that the three-dimensional data encoding device and the three-dimensional data decoding device may encode and decode any one of a dense branch and a sparse branch first. In addition, encoding processes or decoding processes may be performed in parallel using processors etc.

[0709]

Moreover, the three-dimensional data encoding device may encode one of a dense branch and a sparse branch. In addition, the three-dimensional data encoding device may encode part of dense branches. For example, when a specific dense branch includes important information, the three-dimensional data encoding device extracts dense branches and sparse branches from an inputted three-dimensional point cloud, and encodes the dense branch including the important information but does not encode the other dense branches and sparse branches. This enables the three-dimensional data encoding device to append the important information to a stream while reducing an amount of bit. For example, when, between a server and a client, the client sends to the server a transmission request for three-dimensional point cloud information about the surroundings of the client, the server encodes important information about the surroundings of the client as a dense branch and

transmits the important information to the client. This enables the server to transmit the information requested by the client while reducing a network bandwidth.

[0710]

Moreover, the three-dimensional data decoding device may decode one of a dense branch and a sparse branch. In addition, the three-dimensional data decoding device may decode part of dense branches. For example, when a specific dense branch includes important information, the three-dimensional data decoding device decodes the specific dense branch but does not decode other dense branches and sparse branches. This enables the three-dimensional data decoding device to obtain necessary information while reducing a processing load of the decoding process.

[0711]

FIG. 106 is a flowchart of the process of separating three-dimensional points (S1771) illustrated in FIG. 104. First, the three-dimensional data encoding device sets layer L and threshold value TH (S1761). It should be noted that the three-dimensional data encoding device may append information indicating set layer L and threshold value TH, to a bitstream.

[0712]

Next, the three-dimensional data encoding device selects a lead branch in layer L as a current branch (S1762). Then, the three-dimensional data encoding device counts the number of valid leaves of the current branch in layer L (S1763). When the number of the valid leaves of the current branch is greater than threshold value TH (YES in S1764), the three-dimensional data encoding device sets the current branch as a dense branch, and appends layer information and branch information to a bitstream (S1765A). On the other hand, when the number of the valid leaves of the current branch is less than threshold value TH (NO in S1764), the three-dimensional data encoding device sets the current branch as a sparse branch, and appends layer information and branch information to a bitstream (S1766A).

[0713]

When processing of all branches in layer L is not completed (NO in S1767), the three-dimensional data encoding device selects the next branch in layer L as a current branch (S1768). And then, the three-dimensional data encoding device performs step S1763 and the subsequent steps on the selected next current branch. The above-described process is repeated until the processing of all the branches in layer L is completed (YES in S1767).

[0714]

It should be noted that although layer L and threshold value TH are preset in the above description, the present disclosure is not necessarily limited to this. For example, the three-dimensional data encoding device sets different combinations of layer L and threshold value TH, generates a dense branch and a sparse branch using each of the combinations, and encodes the dense branch and the sparse branch. The three-dimensional data encoding device finally encodes the dense branch and the sparse branch using, among the combinations, a combination of layer L and threshold value TH having the highest coding efficiency for encoded data generated. This makes it possible to improve the coding efficiency. Moreover, for example, the three-dimensional data encoding device may calculate layer L and threshold value TH. For example, the three-dimensional data encoding device may set, to layer L, a value half as much as the maximum value of layers included in a tree structure. Furthermore, the three-dimensional data encoding device may set, to threshold value TH, a value half as much as a total number of three-dimensional points included in the tree structure.

[0715]

The following describes an example of a syntax of encoded data of a three-dimensional point cloud according to the present variation. FIG. 107 is a diagram illustrating an example of this syntax. The example of the syntax illustrated in FIG. 107 is obtained by appending layer_id[i] that is layer information and branch_id[i] that is branch information, to the example of the syntax illustrated in FIG. 101.

[0716]

layer_id[i] indicates a layer number of the i-th three-dimensional point sub-cloud. branch_id[i] indicates a branch number in layer_id[i] of the i-th three-dimensional point sub-cloud.

[0717]

layer_id[i] and branch_id[i] are layer information and branch information that indicate, for example, a position of a branch in an octree. For example, layer_id[i] = 2 and branch_id[i] = 5 indicate that the i-th branch is the fifth branch in layer 2.

[0718]

It should be noted that the three-dimensional data encoding device may entropy encode the encoded data generated by the above-described method. For example, the three-dimensional data encoding device binarizes each value and performs arithmetic encoding on the binarized value.

[0719]

Although the example of the quadtree structure or the octree structure has been given in the present variation, the present disclosure is not necessarily limited to this. The above-described method may be applied to an N-ary (N is an integer greater than or equal to 2) tree, such as a binary tree and a hexadecatree, or another tree structure.

[0720]

As stated above, the three-dimensional data encoding device according to the present embodiment performs the process illustrated in FIG. 108.

[0721]

First, the three-dimensional data encoding device generates an N-ary (N is an integer greater than or equal to 2) tree structure of three-dimensional points included in three-dimensional data (S1801).

[0722]

Next, the three-dimensional data encoding device generates first encoded data by

encoding, using a first encoding process, a first branch having, as a root, a first node included in a first layer that is one of layers included in the N-ary tree structure (S1802).

[0723]

In addition, the three-dimensional data encoding device generates second encoded data by encoding, using a second encoding process different from the first encoding process, a second branch having, as a root, a second node that is included in the first layer and different from the first node (S1803).

[0724]

Then, the three-dimensional data encoding device generates a bitstream including the first encoded data and the second encoded data (S1804).

[0725]

Since this enables the three-dimensional data encoding device to apply an encoding process suitable for each branch included in the N-ary tree structure, it is possible to improve the coding efficiency.

[0726]

For example, the number of three-dimensional points included in the first branch is less than a predetermined threshold value, and the number of three-dimensional points included in the second branch is greater than the threshold value. In other words, when the number of three-dimensional points included in a current branch is less than a threshold value, the three-dimensional data encoding device sets the current branch as the first branch, and when the number of three-dimensional points included in the current branch is greater than the threshold value, the three-dimensional data encoding device sets the current branch as the second branch.

[0727]

For example, the first encoded data includes first information indicating that a first N-ary tree structure of first three-dimensional points included in the first branch is expressed using a first formula. The second encoded data includes second information indicating that

a second N-ary tree structure of second three-dimensional points included in the second branch is expressed using a second formula. In other words, the first encoding process and the second encoding process differ in encoding scheme.

[0728]

For example, the location encoding is used in the first encoding process, and the occupancy encoding is used in the second encoding process. In other words, the first information includes pieces of three-dimensional point information each of which is associated with a corresponding one of the first three-dimensional points. Each of the pieces of three-dimensional point information includes an index associated with each of layers in the first N-ary tree structure. Each of the indexes indicates, among N sub-blocks belonging to a corresponding one of the layers, a sub-block to which a corresponding one of the first three-dimensional points belongs. The second information includes pieces of 1-bit information each of which is associated with a corresponding one of sub-blocks belonging to layers in the second N-ary tree structure, and indicates whether a three-dimensional point is present in the corresponding sub-block.

[0729]

For example, a quantization parameter used in the second encoding process is different from a quantization parameter used in the first encoding process. In other words, the first encoding process and the second encoding process are identical in encoding scheme, but differ in parameter for use.

[0730]

For example, as illustrated in FIG. 92 and FIG. 93, in the encoding of the first branch, the three-dimensional data encoding device encodes, using the first encoding process, the tree structure including the first branch and the tree structure from the root of the N-ary tree structure to the first node, and in the encoding of the second branch, the three-dimensional data encoding device encodes, using the second encoding process, the tree structure including the second branch and the tree structure from the root of the N-ary tree

structure to the second node.

[0731]

For example, the first encoded data includes encoded data of the first branch, and third information indicating a position of the first node in the N-ary tree structure. The second encoded data includes encoded data of the second branch, and fourth information indicating a position of the second node in the N-ary tree structure.

[0732]

For example, the third information includes information (layer information) indicating the first layer, and information (branch information) indicating which one of nodes included in the first layer the first node is. The fourth information includes the information (layer information) indicating the first layer, and information (branch information) indicating which one of nodes included in the first layer the second node is.

[0733]

For example, the first encoded data includes information (numPoint) indicating the number of three-dimensional points included in the first branch, and the second encoded data includes information (numPoint) indicating the number of three-dimensional points included in the second branch.

[0734]

For example, the three-dimensional data encoding device includes a processor and memory, and the processor performs the above process using the memory.

[0735]

The three-dimensional data decoding device according to the present embodiment performs the process illustrated in FIG. 109.

[0736]

First, the three-dimensional data decoding device obtains, from a bitstream, first encoded data obtained by encoding a first branch having, as a root, a first node included in a first layer that is one of layers included in an N-ary (N is an integer greater than or equal to 2)

tree structure of three-dimensional points, and second encoded data obtained by encoding a second branch having, as a root, a second node that is included in the first layer and different from the first node (S1811).

[0737]

Next, the three-dimensional data decoding device generates first decoded data of the first branch by decoding the first encoded data using a first decoding process (S1812).

[0738]

In addition, the three-dimensional data decoding device generates second decoded data of the second branch by decoding the second encoded data using a second decoding process different from the first decoding process (S1813).

[0739]

Then, the three-dimensional data decoding device restores three-dimensional points using the first decoded data and the second decoded data (S1814). For example, these three-dimensional points include three-dimensional points indicated by the first decoded data, and three-dimensional points indicated by the second decoded data.

[0740]

This enables the three-dimensional data decoding device to decode the bitstream for which the coding efficiency is improved.

[0741]

For example, the number of three-dimensional points included in the first branch is less than a predetermined threshold value, and the number of three-dimensional points included in the second branch is greater than the threshold value.

[0742]

For example, the first encoded data includes first information indicating that a first N-ary tree structure of first three-dimensional points included in the first branch is expressed using a first formula. The second encoded data includes second information indicating that a second N-ary tree structure of second three-dimensional points included in the second

branch is expressed using a second formula. In other words, the first decoding process and the second decoding process differ in encoding scheme (decoding scheme).

[0743]

For example, the location encoding is used for the first encoded data, and the occupancy encoding is used for the second encoded data. In other words, the first information includes pieces of three-dimensional point information each of which is associated with a corresponding one of the first three-dimensional points. Each of the pieces of three-dimensional point information includes an index associated with each of layers in the first N-ary tree structure. Each of the indexes indicates, among N sub-blocks belonging to a corresponding one of the layers, a sub-block to which a corresponding one of the first three-dimensional points belongs. The second information includes pieces of 1-bit information each of which is associated with a corresponding one of sub-blocks belonging to layers in the second N-ary tree structure, and indicates whether a three-dimensional point is present in the corresponding sub-block.

[0744]

For example, a quantization parameter used in the second decoding process is different from a quantization parameter used in the first decoding process. In other words, the first decoding process and the second decoding process are identical in encoding scheme (decoding scheme), but differ in parameter for use.

[0745]

For example, as illustrated in FIG. 92 and FIG. 93, in the decoding of the first branch, the three-dimensional data decoding device decodes, using the first decoding process, the tree structure including the first branch and the tree structure from the root of the N-ary tree structure to the first node, and in the decoding of the second branch, the three-dimensional data decoding device decodes, using the second decoding process, the tree structure including the second branch and the tree structure from the root of the N-ary tree structure to the second node.

[0746]

For example, the first encoded data includes encoded data of the first branch, and third information indicating a position of the first node in the N-ary tree structure. The second encoded data includes encoded data of the second branch, and fourth information indicating a position of the second node in the N-ary tree structure.

[0747]

For example, the third information includes information (layer information) indicating the first layer, and information (branch information) indicating which one of nodes included in the first layer the first node is. The fourth information includes the information (layer information) indicating the first layer, and information (branch information) indicating which one of nodes included in the first layer the second node is.

[0748]

For example, the first encoded data includes information (numPoint) indicating the number of three-dimensional points included in the first branch, and the second encoded data includes information (numPoint) indicating the number of three-dimensional points included in the second branch.

[0749]

For example, the three-dimensional data decoding device includes a processor and memory, and the processor performs the above process using the memory.

[0750]

EMBODIMENT 11

In the present embodiment, adaptive entropy encoding (arithmetic encoding) performed on occupancy codes of an octree will be described.

[0751]

FIG. 110 is a diagram illustrating an example of a quadtree structure. FIG. 111 is a diagram illustrating occupancy codes of the tree structure illustrated in FIG. 110. FIG. 112 is a diagram schematically illustrating an operation performed by a three-dimensional data

encoding device according to the present embodiment.

[0752]

The three-dimensional data encoding device according to the present embodiment entropy encodes an 8-bit occupancy code in an octree. The three-dimensional data encoding device also updates a coding table in an entropy encoding process for occupancy code. Additionally, the three-dimensional data encoding device does not use a single coding table but uses an adaptive coding table in order to use similarity information of three-dimensional points. In other words, the three-dimensional data encoding device uses coding tables.

[0753]

Similarity information is, for example, geometry information of a three-dimensional point, structure information of an octree, or attribute information of a three-dimensional point.

[0754]

It should be noted that although the quadtree is shown as the example in FIG. 110 to FIG. 112, the same method may be applied to an N-ary tree such as a binary tree, an octree, and a hexadecatree. For example, the three-dimensional data encoding device entropy encodes an 8-bit occupancy code in the case of an octree, a 4-bit occupancy code in the case of a quadtree, and a 16-bit occupancy code in the case of a hexadecatree, using an adaptive table (also referred to as a coding table).

[0755]

The following describes an adaptive entropy encoding process using geometry information of a three-dimensional point.

[0756]

When local geometries of two nodes in a tree structure are similar to each other, there is a chance that occupancy states (i.e., states each indicating whether a three-dimensional point is included) of child nodes are similar to each other. As a result, the three-dimensional data encoding device performs grouping using a local geometry of a parent

node. This enables the three-dimensional data encoding device to group together the occupancy states of the child nodes, and use a different coding table for each group. Accordingly, it is possible to improve the entropy encoding efficiency.

[0757]

FIG. 113 is a diagram illustrating an example of geometry information. Geometry information includes information indicating whether each of neighboring nodes of a current node is occupied (i.e., includes a three-dimensional point). For example, the three-dimensional data encoding device calculates a local geometry of the current node using information indicating whether a neighboring node includes a three-dimensional point (is occupied or non-occupied). A neighboring node is, for example, a node spatially located around a current node, or a node located in the same position in a different time as the current node or spatially located around the position.

[0758]

In FIG. 113, a hatched cube indicates a current node. A white cube is a neighboring node, and indicates a node including a three-dimensional point. In FIG. 113, the geometry pattern indicated in (2) is obtained by rotating the geometry pattern indicated in (1). Accordingly, the three-dimensional data encoding device determines that these geometry patterns have a high geometry similarity, and entropy encodes the geometry patterns using the same coding table. In addition, the three-dimensional data encoding device determines that the geometry patterns indicated in (3) and (4) have a low geometry similarity, and entropy encodes the geometry patterns using other coding tables.

[0759]

FIG. 114 is a diagram illustrating an example of occupancy codes of current nodes in the geometry patterns of (1) to (4) illustrated in FIG. 113, and coding tables used for entropy encoding. As illustrated above, the three-dimensional data encoding device determines that the geometry patterns of (1) and (2) are included in the same geometry group, and uses same coding table A for the geometry patterns of (1) and (2). The three-dimensional data

encoding device uses coding table B and coding table C for the geometry patterns of (3) and (4), respectively.

[0760]

As illustrated in FIG. 114, there is a case in which the occupancy codes of the current nodes in the geometry patterns of (1) and (2) included in the same geometry group are identical to each other.

[0761]

Next, the following describes an adaptive entropy encoding process using structure information of a tree structure. For example, structure information includes information indicating a layer to which a current node belongs.

[0762]

FIG. 115 is a diagram illustrating an example of a tree structure. Generally speaking, a local shape of an object depends on a search criterion. For example, a tree structure tends to be sparser in a lower layer than in an upper layer. Accordingly, the three-dimensional data encoding device uses different coding tables for upper layers and lower layers as illustrated in FIG. 115, which makes it possible to improve the entropy encoding efficiency.

[0763]

In other words, when the three-dimensional data encoding device encodes an occupancy code of each layer, the three-dimensional data encoding device may use a different coding table for each layer. For example, when the three-dimensional data encoding device encodes an occupancy code of layer N (N = 0 to 6), the three-dimensional data encoding device may perform entropy encoding on the tree structure illustrated in FIG. 115 using a coding table for layer N. Since this enables the three-dimensional data encoding device to select a coding table in accordance with an appearance pattern of an occupancy code of each layer, the three-dimensional data encoding device can improve the coding efficiency.

[0764]

Moreover, as illustrated in FIG. 115, the three-dimensional data encoding device may use coding table A for the occupancy codes of layer 0 to layer 2, and may use coding table B for the occupancy codes of layer 3 to layer 6. Since this enables the three-dimensional data encoding device to select a coding table in accordance with an appearance pattern of the occupancy code for each group of layers, the three-dimensional data encoding device can improve the coding efficiency. The three-dimensional data encoding device may append information of the coding table used for each layer, to a header of a bitstream. Alternatively, the coding table used for each layer may be predefined by standards etc.

[0765]

Next, the following describes an adaptive entropy encoding process using attribute information (property information) of a three-dimensional point. For example, attribute information includes information about an object including a current node, or information about a normal vector of the current node.

[0766]

It is possible to group together three-dimensional points having a similar geometry, using pieces of attribute information of the three-dimensional points. For example, a normal vector indicating a direction of each of the three-dimensional points may be used as common attribute information of the three-dimensional points. It is possible to find a geometry relating to a similar occupancy code in a tree structure by using the normal vector.

[0767]

Moreover, a color or a degree of reflection (reflectance) may be used as attribute information. For example, the three-dimensional data encoding device groups together three-dimensional points having a similar geometry, using the colors or reflectances of the three-dimensional points, and performs a process such as switching between coding tables for each of the groups.

[0768]

FIG. 116 is a diagram for describing switching between coding tables based on a normal vector. As illustrated in FIG. 116, when normal vector groups to which normal vectors of current nodes belong are different, different coding tables are used. For example, a normal vector included in a predetermined range is categorized into one normal vector group.

[0769]

When objects belong in different categories, there is a high possibility that occupancy codes are different. Accordingly, the three-dimensional data encoding device may select a coding table in accordance with a category of an object to which a current node belongs. FIG. 117 is a diagram for describing switching between coding tables based on a category of an object. As illustrated in FIG. 117, when objects belong in different categories, different coding tables are used.

[0770]

The following describes an example of a structure of a bitstream according to the present embodiment. FIG. 118 is a diagram illustrating an example of a structure of a bitstream generated by the three-dimensional data encoding device according to the present embodiment. As illustrated in FIG. 118, the bitstream includes a coding table group, table indexes, and encoded occupancy codes. The coding table group includes coding tables.

[0771]

A table index indicates a coding table used for entropy encoding of a subsequent encoded occupancy code. An encoded occupancy code is an occupancy code that has been entropy encoded. As illustrated in FIG. 118, the bitstream also includes combinations of a table index and an encoded occupancy code.

[0772]

For example, in the example illustrated in FIG. 118, encoded occupancy code 0 is data that has been entropy encoded using a context model (also referred to as a context) indicated by table index 0. Encoded occupancy code 1 is data that has been entropy

encoded using a context indicated by table index 1. A context for encoding encoded occupancy code 0 may be predefined by standards etc., and a three-dimensional data decoding device may use this context when decoding encoded occupancy code 0. Since this eliminates the need for appending the table index to the bitstream, it is possible to reduce overhead.

[0773]

Moreover, the three-dimensional data encoding device may append, in the header, information for resetting each context.

[0774]

The three-dimensional data encoding device determines a coding table using geometry information, structure information, or attribute information of a current node, and encodes an occupancy code using the determined coding table. The three-dimensional data encoding device appends a result of the encoding and information (e.g., a table index) of the coding table used for the encoding to a bitstream, and transmits the bitstream to the three-dimensional data decoding device. This enables the three-dimensional data decoding device to decode the occupancy code using the information of the coding table appended to the header.

[0775]

Moreover, the three-dimensional data encoding device need not append information of a coding table used for encoding to a bitstream, and the three-dimensional data decoding device may determine a coding table using geometry information, structure information, or attribute information of a current node that has been decoded, using the same method as the three-dimensional data encoding device, and decode an occupancy code using the determined coding table. Since this eliminates the need for appending the information of the coding table to the bitstream, it is possible to reduce overhead.

[0776]

FIG. 119 and FIG. 120 each are a diagram illustrating an example of a coding table.

As illustrated in FIG. 119 and FIG. 120, one coding table shows, for each value of an 8-bit occupancy code, a context model and a context model type associated with the value.

[0777]

As with the coding table illustrated in FIG. 119, the same context model (context) may be applied to occupancy codes. In addition, a different context model may be assigned to each occupancy code. Since this enables assignment of a context model in accordance with a probability of appearance of an occupancy code, it is possible to improve the coding efficiency.

[0778]

A context model type indicates, for example, whether a context model is a context model that updates a probability table in accordance with an appearance frequency of an occupancy code, or is a context model having a fixed probability table.

[0779]

Next, the following gives another example of a bitstream and a coding table. FIG. 121 is a diagram illustrating a variation of a structure of a bitstream. As illustrated in FIG. 121, the bitstream includes a coding table group and an encoded occupancy code. The coding table group includes coding tables.

[0780]

FIG. 122 and FIG. 123 each are a diagram illustrating an example of a coding table. As illustrated in FIG. 122 and FIG. 123, one coding table shows, for each 1 bit included in an occupancy code, a context model and a context model type associated with the 1 bit.

[0781]

FIG. 124 is a diagram illustrating an example of a relationship between an occupancy code and bit numbers of the occupancy code.

[0782]

As stated above, the three-dimensional data encoding device may handle an occupancy code as binary data, assign a different context model for each bit, and entropy

encode the occupancy code. Since this enables assignment of a context model in accordance with a probability of appearance of each bit of the occupancy code, it is possible to improve the coding efficiency.

[0783]

Specifically, each bit of the occupancy code corresponds to a sub-block obtained by dividing a spatial block corresponding to a current node. Accordingly, when sub-blocks in the same spatial position in a block have the same tendency, it is possible to improve the coding efficiency. For example, when a ground surface or a road surface crosses through a block, in an octree, four lower blocks include three-dimensional points, and four upper blocks include no three-dimensional point. Additionally, the same pattern appears in blocks horizontally arranged. Accordingly, it is possible to improve the coding efficiency by switching between contexts for each bit as described above.

[0784]

A context model that updates a probability table in accordance with an appearance frequency of each bit of an occupancy code may also be used. In addition, a context model having a fixed probability table may be used.

[0785]

Next, the following describes procedures for a three-dimensional data encoding process and a three-dimensional data decoding process according to the present embodiment.

[0786]

FIG. 125 is a flowchart of a three-dimensional data encoding process including an adaptive entropy encoding process using geometry information.

[0787]

In a decomposition process, an octree is generated from an initial bounding box of three-dimensional points. A bounding box is divided in accordance with the position of a three-dimensional point in the bounding box. Specifically, a non-empty sub-space is further divided. Next, information indicating whether a sub-space includes a three-dimensional

point is encoded into an occupancy code. It should be noted that the same process is performed in the processes illustrated in FIG. 127 and FIG. 129.

[0788]

First, the three-dimensional data encoding device obtains inputted three-dimensional points (S1901). Next, the three-dimensional data encoding device determines whether a decomposition process per unit length is completed (S1902).

[0789]

When the decomposition process per unit length is not completed (NO in S1902), the three-dimensional data encoding device generates an octree by performing the decomposition process on a current node (S1903).

[0790]

Then, the three-dimensional data encoding device obtains geometry information (S1904), and selects a coding table based on the obtained geometry information (S1905). Here, as stated above, the geometry information is information indicating, for example, a geometry of occupancy states of neighboring blocks of a current node.

[0791]

After that, the three-dimensional data encoding device entropy encodes an occupancy code of the current node using the selected coding table (S1906).

[0792]

Steps S1903 to S1906 are repeated until the decomposition process per unit length is completed. When the decomposition process per unit length is completed (YES in S1902), the three-dimensional data encoding device outputs a bitstream including generated information (S1907).

[0793]

The three-dimensional data encoding device determines a coding table using geometry information, structure information, or attribute information of a current node, and encodes a bit sequence of an occupancy code using the determined coding table. The

three-dimensional data encoding device appends a result of the encoding and information (e.g., a table index) of the coding table used for the encoding to a bitstream, and transmits the bitstream to the three-dimensional data decoding device. This enables the three-dimensional data decoding device to decode the occupancy code using the information of the coding table appended to the header.

[0794]

Moreover, the three-dimensional data encoding device need not append information of a coding table used for encoding to a bitstream, and the three-dimensional data decoding device may determine a coding table using geometry information, structure information, or attribute information of a current node that has been decoded, using the same method as the three-dimensional data encoding device, and decode an occupancy code using the determined coding table. Since this eliminates the need for appending the information of the coding table to the bitstream, it is possible to reduce overhead.

[0795]

FIG. 126 is a flowchart of a three-dimensional data decoding process including an adaptive entropy decoding process using geometry information.

[0796]

A decomposition process included in the decoding process is similar to the decomposition process included in the above-described encoding process, they differ in the following point. The three-dimensional data decoding device divides an initial bounding box using a decoded occupancy code. When the three-dimensional data decoding device completes a process per unit length, the three-dimensional data decoding device stores the position of a bounding box as the position of a three-dimensional point. It should be noted that the same process is performed in the processes illustrated in FIG. 128 and FIG. 130.

[0797]

First, the three-dimensional data decoding device obtains an inputted bitstream (S1911). Next, the three-dimensional data decoding device determines whether a

decomposition process per unit length is completed (S1912).

[0798]

When the decomposition process per unit length is not completed (NO in S1912), the three-dimensional data decoding device generates an octree by performing the decomposition process on a current node (S1913).

[0799]

Then, the three-dimensional data decoding device obtains geometry information (S1914), and selects a coding table based on the obtained geometry information (S1915). Here, as stated above, the geometry information is information indicating, for example, a geometry of occupancy states of neighboring blocks of a current node.

[0800]

After that, the three-dimensional data decoding device entropy decodes an occupancy code of the current node using the selected coding table (S1916).

[0801]

Steps S1913 to S1916 are repeated until the decomposition process per unit length is completed. When the decomposition process per unit length is completed (YES in S1912), the three-dimensional data decoding device outputs three-dimensional points (S1917).

[0802]

FIG. 127 is a flowchart of a three-dimensional data encoding process including an adaptive entropy encoding process using structure information.

[0803]

First, the three-dimensional data encoding device obtains inputted three-dimensional points (S1921). Next, the three-dimensional data encoding device determines whether a decomposition process per unit length is completed (S1922).

[0804]

When the decomposition process per unit length is not completed (NO in S1922), the three-dimensional data encoding device generates an octree by performing the

decomposition process on a current node (S1923).

[0805]

Then, the three-dimensional data encoding device obtains structure information (S1924), and selects a coding table based on the obtained structure information (S1925). Here, as stated above, the structure information is information indicating, for example, a layer to which a current node belongs.

[0806]

After that, the three-dimensional data encoding device entropy encodes an occupancy code of the current node using the selected coding table (S1926).

[0807]

Steps S1923 to S1926 are repeated until the decomposition process per unit length is completed. When the decomposition process per unit length is completed (YES in S1922), the three-dimensional data encoding device outputs a bitstream including generated information (S1927).

[0808]

FIG. 128 is a flowchart of a three-dimensional data decoding process including an adaptive entropy decoding process using structure information.

[0809]

First, the three-dimensional data decoding device obtains an inputted bitstream (S1931). Next, the three-dimensional data decoding device determines whether a decomposition process per unit length is completed (S1932).

[0810]

When the decomposition process per unit length is not completed (NO in S1932), the three-dimensional data decoding device generates an octree by performing the decomposition process on a current node (S1933).

[0811]

Then, the three-dimensional data decoding device obtains structure information

(S1934), and selects a coding table based on the obtained structure information (S1935). Here, as stated above, the structure information is information indicating, for example, a layer to which a current node belongs.

[0812]

After that, the three-dimensional data decoding device entropy decodes an occupancy code of the current node using the selected coding table (S1936).

[0813]

Steps S1933 to S1936 are repeated until the decomposition process per unit length is completed. When the decomposition process per unit length is completed (YES in S1932), the three-dimensional data decoding device outputs three-dimensional points (S1937).

[0814]

FIG. 129 is a flowchart of a three-dimensional data encoding process including an adaptive entropy encoding process using attribute information.

[0815]

First, the three-dimensional data encoding device obtains inputted three-dimensional points (S1941). Next, the three-dimensional data encoding device determines whether a decomposition process per unit length is completed (S1942).

[0816]

When the decomposition process per unit length is not completed (NO in S1942), the three-dimensional data encoding device generates an octree by performing the decomposition process on a current node (S1943).

[0817]

Then, the three-dimensional data encoding device obtains attribute information (S1944), and selects a coding table based on the obtained attribute information (S1945). Here, as stated above, the attribute information is information indicating, for example, a normal vector of a current node.

[0818]

After that, the three-dimensional data encoding device entropy encodes an occupancy code of the current node using the selected coding table (S1946).

[0819]

Steps S1943 to S1946 are repeated until the decomposition process per unit length is completed. When the decomposition process per unit length is completed (YES in S1942), the three-dimensional data encoding device outputs a bitstream including generated information (S1947).

[0820]

FIG. 130 is a flowchart of a three-dimensional data decoding process including an adaptive entropy decoding process using attribute information.

[0821]

First, the three-dimensional data decoding device obtains an inputted bitstream (S1951). Next, the three-dimensional data decoding device determines whether a decomposition process per unit length is completed (S1952).

[0822]

When the decomposition process per unit length is not completed (NO in S1952), the three-dimensional data decoding device generates an octree by performing the decomposition process on a current node (S1953).

[0823]

Then, the three-dimensional data encoding device obtains attribute information (S1954), and selects a coding table based on the obtained attribute information (S1955). Here, as stated above, the attribute information is information indicating, for example, a normal vector of a current node.

[0824]

After that, the three-dimensional data decoding device entropy decodes an occupancy code of the current node using the selected coding table (S1956).

[0825]

Steps S1953 to S1956 are repeated until the decomposition process per unit length is completed. When the decomposition process per unit length is completed (YES in S1952), the three-dimensional data decoding device outputs three-dimensional points (S1957).

[0826]

FIG. 131 is a flowchart of the process of selecting a coding table using geometry information (S1905).

[0827]

The three-dimensional data encoding device may select a coding table to be used for entropy encoding of an occupancy code, using, as geometry information, information of a geometry group of a tree structure, for example. Here, information of a geometry group is information indicating a geometry group including a geometry pattern of a current node.

[0828]

As illustrated in FIG. 131, when a geometry group indicated by geometry information is geometry group 0 (YES in S1961), the three-dimensional data encoding device selects coding table 0 (S1962). When the geometry group indicated by the geometry information is geometry group 1 (YES in S1963), the three-dimensional data encoding device selects coding table 1 (S1964). In any other case (NO in S1963), the three-dimensional data encoding device selects coding table 2 (S1965).

[0829]

It should be noted that a method of selecting a coding table is not limited to the above. For example, when a geometry group indicated by geometry information is geometry group 2, the three-dimensional data encoding device may further select a coding table according to a value of the geometry group, such as using coding table 2.

[0830]

For example, a geometry group is determined using occupancy information indicating whether a node neighboring a current node includes a point cloud. Geometry patterns that become the same shape by transform such as rotation being applied to may be

included in the same geometry group. The three-dimensional data encoding device may select a geometry group using occupancy information of a node that neighbors a current node or is located around the current node, and belongs to the same layer as the current node. In addition, the three-dimensional data encoding device may select a geometry group using occupancy information of a node that belongs to a layer different from that of a current node. For example, the three-dimensional data encoding device may select a geometry group using occupancy information of a parent node, a node neighboring the parent node, or a node located around the parent node.

[0831]

It should be noted that the same applies to the process of selecting a coding table using geometry information (S1915) in the three-dimensional data decoding device.

[0832]

FIG. 132 is a flowchart of the process of selecting a coding table using structure information (S1925).

[0833]

The three-dimensional data encoding device may select a coding table to be used for entropy encoding of an occupancy code, using, as structure information, layer information of a tree structure, for example. Here, the layer information indicates, for example, a layer to which a current node belongs.

[0834]

As illustrated in FIG. 132, when a current node belongs to layer 0 (YES in S1971), the three-dimensional data encoding device selects coding table 0 (S1972). When the current node belongs to layer 1 (YES in S1973), the three-dimensional data encoding device selects coding table 1 (S1974). In any other case (NO in S1973), the three-dimensional data encoding device selects coding table 2 (S1975).

[0835]

It should be noted that a method of selecting a coding table is not limited to the

above. For example, when a current node belongs to layer 2, the three-dimensional data encoding device may further select a coding table in accordance with the layer to which the current node belongs, such as using coding table 2.

[0836]

The same applies to the process of selecting a coding table using structure information (S1935) in the three-dimensional data decoding device.

[0837]

FIG. 133 is a flowchart of the process of selecting a coding table using attribute information (S1945).

[0838]

The three-dimensional data encoding device may select a coding table to be used for entropy encoding of an occupancy code, using, as attribute information, information about an object to which a current node belongs or information about a normal vector of the current node.

[0839]

As illustrated in FIG. 133, when a normal vector of a current node belongs to normal vector group 0 (YES in S1981), the three-dimensional data encoding device selects coding table 0 (S1982). When the normal vector of the current node belongs to normal vector group 1 (YES in S1983), the three-dimensional data encoding device selects coding table 1 (S1984). In any other case (NO in S1983), the three-dimensional data encoding device selects coding table 2 (S1985).

[0840]

It should be noted that a method of selecting a coding table is not limited to the above. For example, when a normal vector of a current node belongs to normal vector group 2, the three-dimensional data encoding device may further select a coding table in accordance with a normal vector group to which the normal vector of the current belongs, such as using coding table 2.

[0841]

For example, the three-dimensional data encoding device selects a normal vector group using information about a normal vector of a current node. For example, the three-dimensional data encoding device determines, as the same normal vector group, normal vectors having a distance between normal vectors that is less than or equal to a predetermined threshold value.

[0842]

The information about the object to which the current node belongs may be information about, for example, a person, a vehicle, or a building.

[0843]

The following describes configurations of three-dimensional data encoding device 1900 and three-dimensional data decoding device 1910 according to the present embodiment. FIG. 134 is a block diagram of three-dimensional data encoding device 1900 according to the present embodiment. Three-dimensional data encoding device 1900 illustrated in FIG. 134 includes octree generator 1901, similarity information calculator 1902, coding table selector 1903, and entropy encoder 1904.

[0844]

Octree generator 1901 generates, for example, an octree from inputted three-dimensional points, and generates an occupancy code for each node included in the octree. Similarity information calculator 1902 obtains, for example, similarity information that is geometry information, structure information, or attribute information of a current node. Coding table selector 1903 selects a context to be used for entropy encoding of an occupancy code, according to the similarity information of the current node. Entropy encoder 1904 generates a bitstream by entropy encoding the occupancy code using the selected context. It should be noted that entropy encoder 1904 may append, to the bitstream, information indicating the selected context.

[0845]

FIG. 135 is a block diagram of three-dimensional data decoding device 1910 according to the present embodiment. Three-dimensional data decoding device 1910 illustrated in FIG. 135 includes octree generator 1911, similarity information calculator 1912, coding table selector 1913, and entropy decoder 1914.

[0846]

Octree generator 1911 generates an octree in order from, for example, a lower layer to an upper layer using information obtained from entropy decoder 1914. Similarity information calculator 1912 obtains similarity information that is geometry information, structure information, or attribute information of a current node. Coding table selector 1913 selects a context to be used for entropy encoding of an occupancy code, according to the similarity information of the current node. Entropy decoder 1914 generates three-dimensional points by entropy decoding the occupancy code using the selected context. It should be noted that entropy decoder 1914 may obtain, by performing decoding, information of the selected context appended to a bitstream, and use the context indicated by the information.

[0847]

As illustrated in FIG. 122 to FIG. 124 above, the contexts are provided to the respective bits of the occupancy code. In other words, the three-dimensional data encoding device entropy encodes a bit sequence representing an N-ary (N is an integer greater than or equal to 2) tree structure of three-dimensional points included in three-dimensional data, using a coding table selected from coding tables. The bit sequence includes N-bit information for each node in the N-ary tree structure. The N-bit information includes N pieces of 1-bit information each indicating whether a three-dimensional point is present in a corresponding one of N child nodes of a corresponding node. In each of the coding tables, a context is provided to each bit of the N-bit information. The three-dimensional data encoding device entropy encodes each bit of the N-bit information using the context provided to the bit in the selected coding table.

[0848]

This enables the three-dimensional data encoding device to improve the coding efficiency by selecting a context for each bit.

[0849]

For example, in the entropy encoding, the three-dimensional data encoding device selects a coding table to be used from coding tables, based on whether a three-dimensional point is present in each of neighboring nodes of a current node. This enables the three-dimensional data encoding device to improve the coding efficiency by selecting a coding table based on whether the three-dimensional point is present in the neighboring node.

[0850]

For example, in the entropy encoding, the three-dimensional data encoding device (i) selects a coding table based on an arrangement pattern indicating an arranged position of a neighboring node in which a three-dimensional point is present, among neighboring nodes, and (ii) selects the same coding table for arrangement patterns that become identical by rotation, among arrangement patterns. This enables the three-dimensional data encoding device to reduce an increase in the number of coding tables.

[0851]

For example, in the entropy encoding, the three-dimensional data encoding device selects a coding table to be used from coding tables, based on a layer to which a current node belongs. This enables the three-dimensional data encoding device to improve the coding efficiency by selecting a coding table based on the layer to which the current node belongs.

[0852]

For example, in the entropy encoding, the three-dimensional data encoding device selects a coding table to be used from coding tables, based on a normal vector of a current node. This enables the three-dimensional data encoding device to improve the coding efficiency by selecting a coding table based on the normal vector.

[0853]

For example, the three-dimensional data encoding device includes a processor and memory, and the processor performs the above process using the memory.

[0854]

The three-dimensional data decoding device entropy decodes a bit sequence representing an N-ary (N is an integer greater than or equal to 2) tree structure of three-dimensional points included in three-dimensional data, using a coding table selected from coding tables. The bit sequence includes N-bit information for each node in the N-ary tree structure. The N-bit information includes N pieces of 1-bit information each indicating whether a three-dimensional point is present in a corresponding one of N child nodes of a corresponding node. In each of the coding tables, a context is provided to each bit of the N-bit information. The three-dimensional data decoding device entropy decodes each bit of the N-bit information using the context provided to the bit in the selected coding table.

[0855]

This enables the three-dimensional data decoding device to improve the coding efficiency by selecting a context for each bit.

[0856]

For example, in the entropy decoding, the three-dimensional data decoding device selects a coding table to be used from coding tables, based on whether a three-dimensional point is present in each of neighboring nodes of a current node. This enables the three-dimensional data decoding device to improve the coding efficiency by selecting a coding table based on whether the three-dimensional point is present in the neighboring node.

[0857]

For example, in the entropy decoding, the three-dimensional data decoding device (i) selects a coding table based on an arrangement pattern indicating an arranged position of a neighboring node in which a three-dimensional point is present, among neighboring nodes, and (ii) selects the same coding table for arrangement patterns that become identical by rotation, among arrangement patterns. This enables the three-dimensional data decoding

device to reduce an increase in the number of coding tables.

[0858]

For example, in the entropy decoding, the three-dimensional data decoding device selects a coding table to be used from coding tables, based on a layer to which a current node belongs. This enables the three-dimensional data decoding device to improve the coding efficiency by selecting a coding table based on the layer to which the current node belongs.

[0859]

For example, in the entropy decoding, the three-dimensional data decoding device selects a coding table to be used from coding tables, based on a normal vector of a current node. This enables the three-dimensional data decoding device to improve the coding efficiency by selecting a coding table based on the normal vector.

[0860]

For example, the three-dimensional data decoding device includes a processor and memory, and the processor performs the above process using the memory.

[0861]

EMBODIMENT 12

In the present embodiment, a method of controlling reference when an occupancy code is encoded will be described. It should be noted that although the following mainly describes an operation of a three-dimensional data encoding device, a three-dimensional data decoding device may perform the same process.

[0862]

FIG. 136 and FIG. 137 each are a diagram illustrating a reference relationship according to the present embodiment. Specifically, FIG. 136 is a diagram illustrating a reference relationship in an octree structure, and FIG. 137 is a diagram illustrating a reference relationship in a spatial region.

[0863]

In the present embodiment, when the three-dimensional data encoding device

encodes encoding information of a current node to be encoded (hereinafter referred to as a current node), the three-dimensional data encoding device refers to encoding information of each node in a parent node to which the current node belongs. In this regard, however, the three-dimensional data encoding device does not refer to encoding information of each node in another node (hereinafter referred to as a parent neighbor node) that is in the same layer as the parent node. In other words, the three-dimensional data encoding device disables or prohibits reference to a parent neighbor node.

[0864]

It should be noted that the three-dimensional data encoding device may permit reference to encoding information of a parent node (hereinafter also referred to as a grandparent node) of the parent node. In other words, the three-dimensional data encoding device may encode the encoding information of the current node by reference to the encoding information of each of the grandparent node and the parent node to which the current node belongs.

[0865]

Here, encoding information is, for example, an occupancy code. When the three-dimensional data encoding device encodes the occupancy code of the current node, the three-dimensional data encoding device refers to information (hereinafter referred to as occupancy information) indicating whether a point cloud is included in each node in the parent node to which the current node belongs. To put it in another way, when the three-dimensional data encoding device encodes the occupancy code of the current node, the three-dimensional data encoding device refers to an occupancy code of the parent node. On the other hand, the three-dimensional data encoding device does not refer to occupancy information of each node in a parent neighbor node. In other words, the three-dimensional data encoding device does not refer to an occupancy code of the parent neighbor node. Moreover, the three-dimensional data encoding device may refer to occupancy information of each node in the grandparent node. In other words, the three-dimensional data encoding

device may refer to the occupancy information of each of the parent node and the parent neighbor node.

[0866]

For example, when the three-dimensional data encoding device encodes the occupancy code of the current node, the three-dimensional data encoding device selects a coding table to be used for entropy encoding of the occupancy code of the current node, using the occupancy code of the grandparent node or the parent node to which the current node belongs. It should be noted that the details will be described later. At this time, the three-dimensional data encoding device need not refer to the occupancy code of the parent neighbor node. Since this enables the three-dimensional data encoding device to, when encoding the occupancy code of the current node, appropriately select a coding table according to information of the occupancy code of the parent node or the grandparent node, the three-dimensional data encoding device can improve the coding efficiency. Moreover, by not referring to the parent neighbor node, the three-dimensional data encoding device can suppress a process of checking the information of the parent neighbor node and reduce a memory capacity for storing the information. Furthermore, scanning the occupancy code of each node of the octree in a depth-first order makes encoding easy.

[0867]

The following describes an example of selecting a coding table using an occupancy code of a parent node. FIG. 138 is a diagram illustrating an example of a current node and neighboring reference nodes. FIG. 139 is a diagram illustrating a relationship between a parent node and nodes. FIG. 140 is a diagram illustrating an example of an occupancy code of the parent node. Here, a neighboring reference node is a node referred to when a current node is encoded, among nodes spatially neighboring the current node. In the example shown in FIG. 138, the neighboring nodes belong to the same layer as the current node. Moreover, node X neighboring the current node in the x direction, node Y neighboring the current block in the y direction, and node Z neighboring the current block in the z direction

are used as the reference neighboring nodes. In other words, one neighboring node is set as a reference neighboring node in each of the x, y, and z directions.

[0868]

It should be noted that the node numbers shown in FIG. 139 are one example, and a relationship between node numbers and node positions is not limited to the relationship shown in FIG. 139. Although node 0 is assigned to the lowest-order bit and node 7 is assigned to the highest-order bit in FIG. 140, assignments may be made in reverse order. In addition, each node may be assigned to any bit.

[0869]

The three-dimensional data encoding device determines a coding table to be used when the three-dimensional data encoding device entropy encodes an occupancy code of a current node, using the following equation, for example.

[0870]

$$CodingTable = (FlagX << 2) + (FlagY << 1) + (FlagZ)$$

[0871]

Here, CodingTable indicates a coding table for an occupancy code of a current node, and indicates one of values ranging from 0 to 7. FlagX is occupancy information of neighboring node X. FlagX indicates 1 when neighboring node X includes a point cloud (is occupied), and indicates 0 when it does not. FlagY is occupancy information of neighboring node Y. FlagY indicates 1 when neighboring node Y includes a point cloud (is occupied), and indicates 0 when it does not. FlagZ is occupancy information of neighboring node Z. FlagZ indicates 1 when neighboring node Z includes a point cloud (is occupied), and indicates 0 when it does not.

[0872]

It should be noted that since information indicating whether a neighboring node is occupied is included in an occupancy code of a parent node, the three-dimensional data encoding device may select a coding table using a value indicated by the occupancy code of

the parent node.

[0873]

From the foregoing, the three-dimensional data encoding device can improve the coding efficiency by selecting a coding table using the information indicating whether the neighboring node of the current node includes a point cloud.

[0874]

Moreover, as illustrated in FIG. 138, the three-dimensional data encoding device may select a neighboring reference node according to a spatial position of the current node in the parent node. In other words, the three-dimensional data encoding device may select a neighboring node to be referred to from the neighboring nodes, according to the spatial position of the current node in the parent node.

[0875]

Next, the following describes examples of configurations of the three-dimensional data encoding device and the three-dimensional data decoding device. FIG. 141 is a block diagram of three-dimensional data encoding device 2100 according to the present embodiment. Three-dimensional data encoding device 2100 illustrated in FIG. 141 includes octree generator 2101, geometry information calculator 2102, coding table selector 2103, and entropy encoder 2104.

[0876]

Octree generator 2101 generates, for example, an octree from inputted three-dimensional points (a point cloud), and generates an occupancy code for each node included in the octree. Geometry information calculator 2102 obtains occupancy information indicating whether a neighboring reference node of a current node is occupied. For example, geometry information calculator 2102 obtains the occupancy information of the neighboring reference node from an occupancy code of a parent node to which the current node belongs. It should be noted that, as illustrated in FIG. 138, geometry information calculator 2102 may select a neighboring reference node according to a position of the

current node in the parent node. In addition, geometry information calculator 2102 does not refer to occupancy information of each node in a parent neighbor node.

[0877]

Coding table selector 2103 selects a coding table to be used for entropy encoding of an occupancy code of the current node, using the occupancy information of the neighboring reference node calculated by geometry information calculator 2102. Entropy encoder 2104 generates a bitstream by entropy encoding the occupancy code using the selected coding table. It should be noted that entropy encoder 2104 may append, to the bitstream, information indicating the selected coding table.

[0878]

FIG. 142 is a block diagram of three-dimensional data decoding device 2110 according to the present embodiment. Three-dimensional data decoding device 2110 illustrated in FIG. 142 includes octree generator 2111, geometry information calculator 2112, coding table selector 2113, and entropy decoder 2114.

[0879]

Octree generator 2111 generates an octree of a space (nodes) using header information of a bitstream etc. Octree generator 2111 generates an octree by, for example, generating a large space (a root node) using the size of a space along the x-axis, y-axis, and z-axis directions appended to the header information, and generating eight small spaces A (nodes A0 to A7) by dividing the space into two along each of the x-axis, y-axis, and z-axis directions. Nodes A0 to A7 are set as a current node in sequence.

[0880]

Geometry information calculator 2112 obtains occupancy information indicating whether a neighboring reference node of a current node is occupied. For example, geometry information calculator 2112 obtains the occupancy information of the neighboring reference node from an occupancy code of a parent node to which the current node belongs. It should be noted that, as illustrated in FIG. 138, geometry information calculator 2112 may select a

neighboring reference node according to a position of the current node in the parent node. In addition, geometry information calculator 2112 does not refer to occupancy information of each node in a parent neighboring node.

[0881]

Coding table selector 2113 selects a coding table (a decoding table) to be used for entropy decoding of the occupancy code of the current node, using the occupancy information of the neighboring reference node calculated by geometry information calculator 2112. Entropy decoder 2114 generates three-dimensional points by entropy decoding the occupancy code using the selected coding table. It should be noted that coding table selector 2113 may obtain, by performing decoding, information of the selected coding table appended to the bitstream, and entropy decoder 2114 may use a coding table indicated by the obtained information.

[0882]

Each bit of the occupancy code (8 bits) included in the bitstream indicates whether a corresponding one of eight small spaces A (nodes A0 to A7) includes a point cloud. Furthermore, the three-dimensional data decoding device generates an octree by dividing small space node A0 into eight small spaces B (nodes B0 to B7), and obtains information indicating whether each node of small space B includes a point cloud, by decoding the occupancy code. In this manner, the three-dimensional data decoding device decodes the occupancy code of each node while generating an octree by dividing a large space into small spaces.

[0883]

The following describes procedures for processes performed by the three-dimensional data encoding device and the three-dimensional data decoding device. FIG. 143 is a flowchart of a three-dimensional data encoding process in the three-dimensional data encoding device. First, the three-dimensional data encoding device determines (defines) a space (a current node) including part or whole of an inputted three-dimensional

point cloud (S2101). Next, the three-dimensional data encoding device generates eight small spaces (nodes) by dividing the current node into eight (S2102). Then, the three-dimensional data encoding device generates an occupancy code for the current node according to whether each node includes a point cloud (S2103).

[0884]

After that, the three-dimensional data encoding device calculates (obtains) occupancy information of a neighboring reference node of the current node from an occupancy code of a parent node of the current node (S2104). Next, the three-dimensional data encoding device selects a coding table to be used for entropy encoding, based on the calculated occupancy information of the neighboring reference node of the current node (S2105). Then, the three-dimensional data encoding device entropy encodes the occupancy code of the current node using the selected coding table (S2106).

[0885]

Finally, the three-dimensional data encoding device repeats a process of dividing each node into eight and encoding an occupancy code of the node, until the node cannot be divided (S2107). In other words, steps S2102 to S2106 are recursively repeated.

[0886]

FIG. 144 is a flowchart of a three-dimensional data decoding process in the three-dimensional data decoding device. First, the three-dimensional data decoding device determines (defines) a space (a current node) to be decoded, using header information of a bitstream (S2111). Next, the three-dimensional data decoding device generates eight small spaces (nodes) by dividing the current node into eight (S2112). Then, the three-dimensional data decoding device calculates (obtains) occupancy information of a neighboring reference node of the current node from an occupancy code of a parent node of the current node (S2113).

[0887]

After that, the three-dimensional data decoding device selects a coding table to be

used for entropy decoding, based on the occupancy information of the neighboring reference node (S2114). Next, the three-dimensional data decoding device entropy decodes the occupancy code of the current node using the selected coding table (S2115).

[0888]

Finally, the three-dimensional data decoding device repeats a process of dividing each node into eight and decoding an occupancy code of the node, until the node cannot be divided (S2116). In other words, steps S2112 to S2115 are recursively repeated.

[0889]

Next, the following describes an example of selecting a coding table. FIG. 145 is a diagram illustrating an example of selecting a coding table. For example, as in coding table 0 shown in FIG. 145, the same context mode may be applied to occupancy codes. Moreover, a different context model may be assigned to each occupancy code. Since this enables assignment of a context model in accordance with a probability of appearance of an occupancy code, it is possible to improve the coding efficiency. Furthermore, a context mode that updates a probability table in accordance with an appearance frequency of an occupancy code may be used. Alternatively, a context model having a fixed probability table may be used.

[0890]

It should be noted that although the coding tables illustrated in FIG. 119 and FIG. 120 are used in the example shown in FIG. 145, the coding tables illustrated in FIG. 122 and FIG. 123 may be used instead.

[0891]

Hereinafter, Variation 1 of the present embodiment will be described. FIG. 146 is a diagram illustrating a reference relationship in the present variation. Although the three-dimensional data encoding device does not refer to the occupancy code of the parent neighbor node in the above-described embodiment, the three-dimensional data encoding device may switch whether to refer to an occupancy code of a parent neighbor node,

according to a specific condition.

[0892]

For example, when the three-dimensional data encoding device encodes an octree while scanning the octree breadth-first, the three-dimensional data encoding device encodes an occupancy code of a current node by reference to occupancy information of a node in a parent neighbor node. In contrast, when the three-dimensional data encoding device encodes the octree while scanning the octree depth-first, the three-dimensional data encoding device prohibits reference to the occupancy information of the node in the parent neighbor node. By appropriately selecting a referable node according to the scan order (encoding order) of nodes of the octree in the above manner, it is possible to improve the coding efficiency and reduce the processing load.

[0893]

It should be noted that the three-dimensional data encoding device may append, to a header of a bitstream, information indicating, for example, whether an octree is encoded breadth-first or depth-first. FIG. 147 is a diagram illustrating an example of a syntax of the header information in this case. octree_scan_order shown in FIG. 147 is encoding order information (an encoding order flag) indicating an encoding order for an octree. For example, when octree_scan_order is 0, breadth-first is indicated, and when octree_scan_order is 1, depth-first is indicated. Since this enables the three-dimensional data decoding device to determine whether a bitstream has been encoded breadth-first or depth-first by reference to octree_scan_order, the three-dimensional data decoding device can appropriately decode the bitstream

[0894]

Moreover, the three-dimensional data encoding device may append, to header information of a bitstream, information indicating whether to prohibit reference to a parent neighbor node. FIG. 148 is a diagram illustrating an example of a syntax of the header information in this case. limit_refer_flag is prohibition switch information (a prohibition

switch flag) indicating whether to prohibit reference to a parent neighbor node. For example, when limit_refer_flag is 1, prohibition of reference to the parent neighbor node is indicated, and when limit_refer_flag is 0, no reference limitation (permission of reference to the parent neighbor node) is indicated.

[0895]

In other words, the three-dimensional data encoding device determines whether to prohibit the reference to the parent neighbor node, and selects whether to prohibit or permit the reference to the parent neighbor node, based on a result of the above determination. In addition, the three-dimensional data encoding device generates a bitstream including prohibition switch information that indicates the result of the determination and indicates whether to prohibit the reference to the parent neighbor node.

[0896]

The three-dimensional data decoding device obtains, from a bitstream, prohibition switch information indicating whether to prohibit reference to a parent neighbor node, and selects whether to prohibit or permit the reference to the parent neighbor node, based on the prohibition switch information.

[0897]

This enables the three-dimensional data encoding device to control the reference to the parent neighbor node and generate the bitstream. That also enables the three-dimensional data decoding device to obtain, from the header of the bitstream, the information indicating whether to prohibit the reference to the parent neighbor node.

[0898]

Although the process of encoding an occupancy code has been described as an example of an encoding process in which reference to a parent neighbor node is prohibited in the present embodiment, the present disclosure is not necessarily limited to this. For example, the same method can be applied when other information of a node of an octree is encoded. For example, the method of the present embodiment may be applied when other

attribute information, such as a color, a normal vector, or a degree of reflection, added to a node is encoded. Additionally, the same method can be applied when a coding table or a predicted value is encoded.

[0899]

Hereinafter, Variation 2 of the present embodiment will be described. In the above description, as illustrated in FIG. 138, the example in which the three reference neighboring nodes are used is given, but four or more reference neighboring nodes may be used. FIG. 149 is a diagram illustrating an example of a current node and neighboring reference nodes.

[0900]

For example, the three-dimensional data encoding device calculates a coding table to be used when the three-dimensional data encoding device entropy encodes an occupancy code of the current node shown in FIG. 149, using the following equation.

[0901]

$$CodingTable = (FlagX0 << 3) + (FlagX1 << 2) + (FlagY << 1) + (FlagZ)$$

[0902]

Here, CodingTable indicates a coding table for an occupancy code of a current node, and indicates one of values ranging from 0 to 15. FlagXN is occupancy information of neighboring node XN (N = 0.. 1). FlaxXN indicates 1 when neighboring node XN includes a point cloud (is occupied), and indicates 0 when it does not. FlagY is occupancy information of neighboring node Y. FlagY indicates 1 when neighboring node Y includes a point cloud (is occupied), and indicates 0 when it does not. FlagZ is occupancy information of neighboring node Z. FlagZ indicates 1 when neighboring node Z includes a point cloud (is occupied), and indicates 0 when it does not.

[0903]

At this time, when a neighboring node, for example, neighboring node X0 in FIG. 149, is unreferable (prohibited from being referred to), the three-dimensional data encoding device may use, as a substitute value, a fixed value such as 1 (occupied) or 0 (unoccupied).

[0904]

FIG. 150 is a diagram illustrating an example of a current node and neighboring reference nodes. As illustrated in FIG. 150, when a neighboring node is unreferable (prohibited from being referred to), occupancy information of the neighboring node may be calculated by reference to an occupancy code of a grandparent node of the current node. For example, the three-dimensional data encoding device may calculate FlagX0 in the above equation using occupancy information of neighboring node G0 instead of neighboring node X0 illustrated in FIG. 150, and may determine a value of a coding table using calculated FlagX0. It should be noted that neighboring node G0 illustrated in FIG. 150 is a neighboring node occupancy or unoccupancy of which can be determined using the occupancy code of the grandparent node. Neighboring node X1 is a neighboring node occupancy or unoccupancy of which can be determined using an occupancy code of a parent node.

[0905]

Hereinafter, Variation 3 of the present embodiment will be described. FIG. 151 and FIG. 152 each are a diagram illustrating a reference relationship according to the present variation. Specifically, FIG. 151 is a diagram illustrating a reference relationship in an octree structure, and FIG. 152 is a diagram illustrating a reference relationship in a spatial region.

[0906]

In the present variation, when the three-dimensional data encoding device encodes encoding information of a current node to be encoded (hereinafter referred to as current node 2), the three-dimensional data encoding device refers to encoding information of each node in a parent node to which current node 2 belongs. In other words, the three-dimensional data encoding device permits reference to information (e.g., occupancy information) of a child node of a first node, among neighboring nodes, that has the same parent node as a current node. For example, when the three-dimensional data encoding device encodes an

occupancy code of current node 2 illustrated in FIG. 151, the three-dimensional data encoding device refers to an occupancy code of a node in the parent node to which current node 2 belongs, for example, the current node illustrated in FIG. 151. As illustrated in FIG. 152, the occupancy code of the current node illustrated in FIG. 151 indicates, for example, whether each node in the current node neighboring current node 2 is occupied. Accordingly, since the three-dimensional data encoding device can select a coding table for the occupancy code of current node 2 in accordance with a more particular shape of the current node, the three-dimensional data encoding device can improve the coding efficiency.

[0907]

The three-dimensional data encoding device may calculate a coding table to be used when the three-dimensional data encoding device entropy encodes the occupancy code of current node 2, using the following equation, for example.

[0908]

CodingTable = (FlagX1 << 5) + (FlagX2 << 4) + (FlagX3 << 3) + (FlagX4 << 2) + (FlagY << 1) + (FlagZ)

[0909]

Here, CodingTable indicates a coding table for an occupancy code of current node 2, and indicates one of values ranging from 0 to 63. FlagXN is occupancy information of neighboring node XN (N = 1.. 4). FlagXN indicates 1 when neighboring node XN includes a point cloud (is occupied), and indicates 0 when it does not. FlagY is occupancy information of neighboring node Y. FlagY indicates 1 when neighboring node Y includes a point cloud (is occupied), and indicates 0 when it does not. FlagZ is occupancy information of neighboring node Z. FlagZ indicates 1 when neighboring node Z includes a point cloud (is occupied), and indicates 0 when it does not.

[0910]

It should be noted that the three-dimensional data encoding device may change a method of calculating a coding table, according to a node position of current node 2 in the

parent node.

[0911]

When reference to a parent neighbor node is not prohibited, the three-dimensional data encoding device may refer to encoding information of each node in the parent neighbor node. For example, when the reference to the parent neighbor node is not prohibited, reference to information (e.g., occupancy information) of a child node of a third node having a different parent node from that of a current node. In the example illustrated in FIG. 150, for example, the three-dimensional data encoding device obtains occupancy information of a child node of neighboring node X0 by reference to an occupancy code of neighboring node X0 having a different parent node from that of the current node. The three-dimensional data encoding device selects a coding table to be used for entropy encoding of an occupancy code of the current node, based on the obtained occupancy information of the child node of neighboring node X0.

[0912]

As stated above, the three-dimensional data encoding device according to the present embodiment encodes information (e.g., an occupancy code) of a current node included in an N-ary tree structure of three-dimensional points included in three-dimensional data, where N is an integer greater than or equal to 2. As illustrated in FIG. 136 and FIG. 137, in the encoding, the three-dimensional data encoding device permits reference to information (e.g., occupancy information) of a first node included in neighboring nodes spatially neighboring the current node, and prohibits reference to information of a second node included in the neighboring nodes, the first node having a same parent node as the current node, the second node having a different parent node from the parent node of the current node. To put it another way, in the encoding, the three-dimensional data encoding device permits reference to information (e.g., an occupancy code) of the parent node, and prohibits reference to information (e.g., an occupancy code) of another node (a parent neighbor node) in the same layer as the parent node.

[0913]

With this, the three-dimensional data encoding device can improve coding efficiency by reference to the information of the first node included in the neighboring nodes spatially neighboring the current node, the first node having the same parent node as the current node. Besides, the three-dimensional data encoding device can reduce a processing amount by not reference to the information of the second node included in the neighboring nodes, the second node having a different parent node from the parent node of the current node. In this manner, the three-dimensional data encoding device can not only improve the coding efficiency but also reduce the processing amount.

[0914]

For example, the three-dimensional data encoding device further determines whether to prohibit the reference to the information of the second node. In the encoding, the three-dimensional data encoding device selects whether to prohibit or permit the reference to the information of the second node, based on a result of the determining. Moreover, the three-dimensional data encoding device generates a bit stream including prohibition switch information (e.g., limit_refer_flag shown in FIG. 148) that indicates the result of the determining and indicates whether to prohibit the reference to the information of the second node.

[0915]

With this, the three-dimensional data encoding device can select whether to prohibit the reference to the information of the second node. In addition, a three-dimensional data decoding device can appropriately perform a decoding process using the prohibition switch information.

[0916]

For example, the information of the current node is information (e.g., an occupancy code) that indicates whether a three-dimensional point is present in each of child nodes belonging to the current node. The information of the first node is information (the

occupancy information of the first node) that indicates whether a three-dimensional point is present in the first node. The information of the second node is information (the occupancy information of the second node) that indicates whether a three-dimensional point is present in the second node.

[0917]

For example, in the encoding, the three-dimensional data encoding device selects a coding table based on whether the three-dimensional point is present in the first node, and entropy encodes the information (e.g., the occupancy code) of the current node using the coding table selected.

[0918]

For example, as illustrated in FIG. 151 and FIG. 152, in the encoding, the three-dimensional data encoding device permits reference to information (e.g., occupancy information) of a child node of the first node, the child node being included in the neighboring nodes.

[0919]

With this, since the three-dimensional data encoding device enables reference to more detailed information of a neighboring node, the three-dimensional data encoding device can improve the coding efficiency.

[0920]

For example, as illustrated in FIG. 138, in the encoding, the three-dimensional data encoding device selects a neighboring node to be referred to from the neighboring nodes according to a spatial position of the current node in the parent node.

[0921]

With this, the three-dimensional data encoding device can refer to an appropriate neighboring node according to the spatial position of the current node in the parent node.

[0922]

For example, the three-dimensional data encoding device includes a processor and

memory, and the processor performs the above process using the memory.

[0923]

The three-dimensional data decoding device according to the present embodiment decodes information (e.g., an occupancy code) of a current node included in an N-ary tree structure of three-dimensional points included in three-dimensional data, where N is an integer greater than or equal to 2. As illustrated in FIG. 136 and FIG. 137, in the decoding, the three-dimensional data decoding device permits reference to information (e.g., occupancy information) of a first node included in neighboring nodes spatially neighboring the current node, and prohibits reference to information of a second node included in the neighboring nodes, the first node having a same parent node as the current node, the second node having a different parent node from the parent node of the current node. To put it another way, in the decoding, the three-dimensional data decoding device permits reference to information (e.g., an occupancy code) of the parent node, and prohibits reference to information (e.g., an occupancy code) of another node (a parent neighbor node) in the same layer as the parent node.

[0924]

With this, the three-dimensional data decoding device can improve coding efficiency by reference to the information of the first node included in the neighboring nodes spatially neighboring the current node, the first node having the same parent node as the current node. Besides, the three-dimensional data decoding device can reduce a processing amount by not reference to the information of the second node included in the neighboring nodes, the second node having a different parent node from the parent node of the current node. In this manner, the three-dimensional data decoding device can not only improve the coding efficiency but also reduce the processing amount.

[0925]

For example, the three-dimensional data decoding device further obtains, from a bitstream, prohibition switch information (e.g., limit_refer_flag shown in FIG. 148)

indicating whether to prohibit the reference to the information of the second node. In the decoding, the three-dimensional data decoding device selects whether to prohibit or permit the reference to the information of the second node, based on the prohibition switch information.

[0926]

With this, the three-dimensional data decoding device can appropriately perform a decoding process using the prohibition switch information.

[0927]

For example, the information of the current node is information (e.g., an occupancy code) that indicates whether a three-dimensional point is present in each of child nodes belonging to the current node. The information of the first node is information (the occupancy information of the first node) that indicates whether a three-dimensional point is present in the first node. The information of the second node is information (the occupancy information of the second node) that indicates whether a three-dimensional point is present in the second node.

[0928]

For example, in the decoding, the three-dimensional data encoding device selects a coding table based on whether the three-dimensional point is present in the first node, and entropy decodes the information (e.g., the occupancy code) of the current node using the coding table selected.

[0929]

For example, as illustrated in FIG. 151 and FIG. 152, in the decoding, the three-dimensional data decoding device permits reference to information (e.g., occupancy information) of a child node of the first node, the child node being included in the neighboring nodes.

[0930]

With this, since the three-dimensional data decoding device enables reference to

more detailed information of a neighboring node, the three-dimensional data decoding device can improve the coding efficiency.

[0931]

For example, as illustrated in FIG. 138, in the decoding, the three-dimensional data decoding device selects a neighboring node to be referred to from the neighboring nodes according to a spatial position of the current node in the parent node.

[0932]

With this, the three-dimensional data decoding device can refer to an appropriate neighboring node according to the spatial position of the current node in the parent node.

[0933]

For example, the three-dimensional data decoding device includes a processor and memory, and the processor performs the above process using the memory.

[0934]

EMBODIMENT 13

Although the following mainly describes an operation of a three-dimensional data encoding device, a three-dimensional data decoding device may perform the same process.

[0935]

In the present embodiment, in the case where the three-dimensional data encoding device encodes an inputted three-dimensional point cloud (a point cloud) using an octree structure, when the three-dimensional data encoding device repeats division until each leaf included in an octree has a single three-dimensional point, and performs encoding, the three-dimensional data encoding device appends, to a bitstream, mode information indicating whether each leaf of an octree includes a single three-dimensional point or one or more three-dimensional points. In addition, when the mode information is true (each leaf of the octree includes a single three-dimensional point), the three-dimensional data encoding device does not encode leaf information about the leaf, and when the mode information is false (each leaf of the octree includes one or more three-dimensional points), the three-dimensional data

encoding device encodes the leaf information.

[0936]

Here, the leaf information includes, for example, information indicating how many three-dimensional points a leaf includes, information indicating relative coordinates etc. of a three-dimensional point included in a leaf as illustrated in FIG. 61 to FIG. 67, or both. Accordingly, since it is unnecessary to encode the leaf information for each leaf of the octree when the leaf includes a single three-dimensional point, it is possible to improve coding efficiency. When each leaf of the octree includes one or more three-dimensional points, the three-dimensional data encoding device appropriately encodes and appends the leaf information to the bitstream. This enables the three-dimensional data decoding device to correctly restore the one or more three-dimensional points included in the leaf using the leaf information.

[0937]

FIG. 153 is a diagram illustrating an example of a syntax of header information of a bitstream according to the present embodiment. This header information is, for example, WLD, SPC, or VLM. single_point_per_leaf shown in FIG. 153 is information indicating whether each leaf of an octree includes a single three-dimensional point or one or more three-dimensional points. Here, "each leaf of an octree includes a single three-dimensional point" means that all of the leaves included in the octree each include a single three-dimensional point (i.e., there is no leaf including two or more three-dimensional points). It should be noted that the octree here is, for example, a unit in which mode information is appended, and corresponds to, for example, WLD, SPC, or VLM.

[0938]

Furthermore, "each leaf of an octree includes one or more three-dimensional points" means that at least one of the leaves included in the octree includes two or more three-dimensional points. In other words, some of the leaves may include two or more three-dimensional points, and the remaining leaves may each include a single

three-dimensional point.

[0939]

For example, that a value of mode information is 1 indicates each leaf including a single three-dimensional point, and that a value of mode information is 0 indicates each leaf including one or more three-dimensional points. It should be noted that even when all of the leaves included in the octree each include a single three-dimensional point, the three-dimensional data encoding device may set mode information (single_point_per_leaf) to 0.

[0940]

FIG. 154 is a diagram illustrating a configuration example of an octree when mode information indicates 1. As illustrated in FIG. 154, when the mode information indicates 1, each leaf includes a single three-dimensional point. FIG. 155 is a diagram illustrating a configuration example of an octree when mode information indicates 0. As illustrated in FIG. 155, when the mode information indicates 0, each leaf includes a single three-dimensional point or two or more three-dimensional points.

[0941]

It should be noted that when a leaf includes two or more three-dimensional points, for example, coordinates of the at two or more three-dimensional points included in the leaf are different from each other. Alternatively, the coordinates of the at two or more three-dimensional points included in the leaf are identical, and pieces of attribute information, such as a color or a degree of reflection, (i.e., types of information) are mutually different. Alternatively, both the coordinates of the two or more three-dimensional points included in the leaf and the pieces of attribute information may be mutually different.

[0942]

Next, an example of a syntax of leaf information will be described. It should be noted that although the example in which the mode information is appended to the header of the bitstream has been given above, the mode information need not be appended to the header,

and standards or a profile or level etc. of standards may specify whether each leaf of the octree includes a single three-dimensional point or one or more three-dimensional points. In this case, the three-dimensional data decoding device can correctly restore the bitstream by, for example, determining whether each leaf of the octree includes a single three-dimensional point or one or more three-dimensional points, by reference to standards information included in the bitstream.

[0943]

FIG. 156 is a diagram illustrating an example of a syntax of information of each node included in an octree. isleaf shown in FIG. 156 is a flag indicating whether a node is a leaf. That isleaf is 1 indicates that a node is a leaf. That isleaf is 0 indicates that a node is not a leaf.

[0944]

It should be noted that information indicating whether a node is a leaf need not be appended to a header. In this case, the three-dimensional data decoding device determines whether a node is a leaf using another method. For example, the three-dimensional data decoding device may determine whether each node of the octree is divided into the smallest possible size, and may determine that a node is a leaf when determining that each node is divided into the smallest possible size. This eliminates the need for encoding the flag indicating whether the node is the leaf, which makes it possible to reduce the code amount of the header.

[0945]

num_point_per_leaf shown in FIG. 156 is leaf information and indicates the number of three-dimensional points included in a leaf. num_point_per_leaf is encoded when single_point_per_leaf==0, and is not encoded when single_point_per_leaf==1.

[0946]

It should be noted that the three-dimensional data encoding device may entropy encode num_point_per_leaf. At this time, the three-dimensional data encoding device may

also perform encoding while switching coding tables. For example, the three-dimensional data encoding device may perform arithmetic encoding on the first bit of num_point_per_leaf using coding table A, and may perform arithmetic encoding on the remaining bits using coding table B.

[0947]

As stated above, the three-dimensional data encoding device may append, to the header of the bitstream, the mode information indicating whether each leaf of the octree includes a single three-dimensional point or one or more three-dimensional points, and may select whether to encode the leaf information (the information indicating the number of three-dimensional points included in the leaf) according to the value of the mode information. Besides, the three-dimensional data encoding device may encode, as the leaf information, positional information of the single or one or more three-dimensional points included in the leaf.

[0948]

It should be noted that the three-dimensional data encoding device may entropy encode single_point_per_leaf, isleaf, and num_point_per_leaf generated by the above method. For example, the three-dimensional data encoding device binarizes each value and performs arithmetic encoding on the binarized value.

[0949]

Although the octree structure has been given as an example in the present embodiment, the present disclosure is not necessarily limited to this. The aforementioned procedure may be applied to an N-ary tree structure such as a quadtree and a hexadecatree.

[0950]

Moreover, when the three-dimensional data encoding device encodes, as the leaf information, pieces of positional information of two or more three-dimensional points in the same leaf, the three-dimensional data encoding device may also encode pieces of attribute information (color, degree of reflectance, etc.) of the two or more three-dimensional points.

In this case, the pieces of positional information of the two or more three-dimensional points and the pieces of attribute information of the same may be associated with each other. For example, when the three-dimensional data encoding device encodes, as the leaf information, pieces of positional information of points A and B in the same leaf, the three-dimensional data encoding device may encode pieces of attribute information of points A and B and append the pieces of attribute information to a bitstream. In other words, the leaf information may include the positional information of point A, the attribute information of point A, the positional information of point B, and the attribute information of point B. The pieces of positional information of points A and B may be associated with the pieces of attribute information.

[0951]

Moreover, the three-dimensional data encoding device may round off at least M pieces of positional information of at least M three-dimensional points in the same leaf, and may encode the at least M pieces of positional information as N pieces of positional information of N three-dimensional points, where N is less than M. In this case, the three-dimensional data encoding device may round off at least M pieces of attribute information of the at least M three-dimensional points by, for example, averaging to generate N pieces of attribute information, and may encode the N pieces of attribute information generated. For example, the three-dimensional data encoding device may round off the pieces of positional information of points A and B in the same leaf to a piece of positional information of one point, and may encode the piece of positional information. In this case, the three-dimensional data encoding device may round off the pieces of attribute information of points A and B by, for example, averaging to calculate a piece of attribute information of one point, and may encode the piece of attribute information calculated.

[0952]

The following describes procedures performed by the three-dimensional data encoding device and three-dimensional data decoding device according to the present

embodiment. FIG. 157 is a flowchart of a three-dimensional data encoding process performed by the three-dimensional data encoding device according to the present embodiment. First, the three-dimensional data encoding device determines whether to perform encoding so that each leaf of an octree includes a single three-dimensional point or to perform encoding so that each leaf of the octree includes one or more three-dimensional points (S2201). For example, the three-dimensional data encoding device may determine whether to perform encoding so that each leaf of the octree includes a single three-dimensional point or to perform encoding so that each leaf of the octree includes one or more three-dimensional points, according to whether to lossless encode an inputted three-dimensional point cloud. For example, when the three-dimensional data encoding device lossless encodes the inputted three-dimensional point cloud, the three-dimensional data encoding device determines to perform encoding so that each leaf of the octree includes a single three-dimensional point. Alternatively, the three-dimensional data encoding device may determine whether to perform encoding so that each leaf of the octree includes a single three-dimensional point or to perform encoding so that each leaf of the octree includes one or more three-dimensional points, according to whether all the coordinates of the inputted three-dimensional points are mutually different and whether to perform encoding so that each of the three-dimensional points is included in a different leaf. For example, when the three-dimensional points include three-dimensional points having the same coordinates, the three-dimensional data encoding device may determine to perform encoding so that each leaf of the octree includes one or more three-dimensional points.

[0953]

When the three-dimensional data encoding device determines to perform encoding so that each leaf of the octree includes a single three-dimensional point (YES in S2201), the three-dimensional data encoding device sets mode information to a value indicating that each leaf of the octree includes a single three-dimensional point (single_point_per_leaf=1), and appends the mode information to a header (S2202).

[0954]

When the three-dimensional data encoding device determines to perform encoding so that each leaf of the octree includes one or more three-dimensional points (NO in S2201), the three-dimensional data encoding device sets mode information to a value indicating that each leaf of the octree includes one or more three-dimensional points (single_point_per_leaf=0), and appends the mode information to a header (S2203).

[0955]

Next, the three-dimensional data encoding device generates an octree structure by dividing a root node into an octree (S2204). At this time, when the mode information indicates that each leaf of the octree is to include a single three-dimensional point, the three-dimensional data encoding device generates an octree in which each leaf of the octree includes a single three-dimensional point, and when the mode information indicates that each leaf of the octree is to include one or more three-dimensional points, the three-dimensional data encoding device generates an octree in which each leaf of the octree includes one or more three-dimensional points.

[0956]

Then, the three-dimensional data encoding device selects a current node to be processed and determines whether the current node is a leaf (S2205). When the current node is the leaf (YES in S2205) and the mode information indicates that each leaf of the octree is to include one or more three-dimensional points (single_point_per_leaf=0) (YES in S2206), the three-dimensional data encoding device encodes leaf information indicating, for example, the number of three-dimensional points included in the leaf (S2207).

[0957]

In contrast, when the current node is the leaf (YES in S2205) and the mode information indicates that each leaf of the octree is to include a single three-dimensional point (single_point_per_leaf=1) (NO in S2206), the three-dimensional data encoding device does not encode leaf information indicating, for example, the number of three-dimensional points

included in the leaf.

[0958]

When the current node is the leaf (NO in S2205), the three-dimensional data encoding device encodes an occupancy code of the current node (S2208).

[0959]

When processing of all nodes is not completed (NO in S2209), the three-dimensional data encoding device selects the next current node and performs step S2205 and the subsequent steps on the selected current node. When the processing of all the nodes is completed (YES in S2209), the three-dimensional data encoding device ends the process.

[0960]

FIG. 158 is a flowchart of a three-dimensional data decoding process performed by the three-dimensional data decoding device according to the present embodiment. First, the three-dimensional data decoding device decodes mode information (single_point_per_leaf) in the header of a bitstream (S2211).

[0961]

Next, the three-dimensional data decoding device generates an octree of a space (nodes) using, for example, header information included in the bitstream (S2212). For example, the three-dimensional data decoding device generates a large space (a root node) using the size of a space along the x-axis, y-axis, and z-axis directions appended to the header information. Subsequently, the three-dimensional data decoding device generates an octree by generating eight small spaces (nodes A0 to A7) by dividing the space into two along each of the x-axis, y-axis, and z-axis directions. In a similar way, the three-dimensional data decoding device divides each of nodes A0 to A7 into eight small spaces. Furthermore, the three-dimensional data decoding device performs decoding of an occupancy code of each node and decoding of leaf information in sequence through the process illustrated in FIG. 158.

[0962]

Specifically, the three-dimensional data decoding device selects a current node to be processed and determines whether the current node is a leaf (S2213). When the current node is the leaf (YES in S2213) and the mode information indicates that each leaf of the octree is to include one or more three-dimensional points (single_point_per_leaf=0) (YES in S2214), the three-dimensional data decoding device decodes leaf information indicating, for example, the number of three-dimensional points included in the leaf (S2215).

[0963]

In contrast, when the current node is the leaf (YES in S2213) and the mode information indicates that each leaf of the octree is to include a single three-dimensional point (single_point_per_leaf=1) (NO in S2214), the three-dimensional data decoding device does not decode leaf information indicating, for example, the number of three-dimensional points included in the leaf.

[0964]

When the current node is not the leaf (NO in S2213), the three-dimensional data decoding device decodes an occupancy code of the current node (S2216).

[0965]

When processing of all nodes is not completed (NO in S2217), the three-dimensional data decoding device selects the next current node and performs step S2213 and the subsequent steps on the selected current node. When the processing of all the nodes is completed (YES in S2217), the three-dimensional data decoding device ends the process.

[0966]

The following describes configurations of the three-dimensional data encoding device and three-dimensional data decoding device according to the present embodiment. FIG. 159 is a block diagram illustrating a configuration of three-dimensional data encoding device 2200 according to the present embodiment. Three-dimensional data encoding device

2200 illustrated in FIG. 159 includes octree generator 2201, mode determiner 2202, and entropy encoder 2203.

[0967]

Octree generator 2201 generates, for example, an octree from inputted three-dimensional points (a point cloud), and generates a corresponding one of an occupancy code and leaf information for each node included in the octree. Mode determiner 2202 determines whether to perform encoding so that each leaf of the octree includes a single three-dimensional point or to perform encoding so that each leaf of the octree includes one or more three-dimensional points, and generates mode information indicating a result of the determination. In other words, mode determiner 2202 sets a value of single_point_per_leaf.

[0968]

Entropy encoder 2203 encodes the leaf information according to the mode information to generate a bitstream. Additionally, entropy encoder 2203 appends the leaf information (single_point_per_leaf) to the bitstream.

[0969]

FIG. 160 is a block diagram illustrating a configuration of three-dimensional data decoding device 2210 according to the present embodiment. Three-dimensional data decoding device 2210 illustrated in FIG. 160 includes octree generator 2211, mode information decoder 2212, and entropy decoder 2213.

[0970]

Octree generator 2211 generates an octree of a space (nodes) using, for example, header information of a bitstream. For example, octree generator 2211 generates a large space (a root node) using the size of a space along the x-axis, y-axis, and z-axis directions appended to the header information, and generates an octree by generating eight small spaces (nodes A0 to A7) by dividing the space into two along each of the x-axis, y-axis, and z-axis directions. In a similar way, octree generator 2211 divides each of nodes A0 to A7 into eight small spaces. As stated above, octree generator 2211 repeats the generation of an octree.

[0971]

Mode information decoder 2212 decodes mode information (single_point_per_leaf) from the header information of the bitstream. It should be noted that mode information decoder 2212 may be included in entropy decoder 2213.

[0972]

Entropy decoder 2213 decodes an occupancy code and leaf information according to the mode information decoded, and generates three-dimensional points using the occupancy code and the leaf information decoded.

[0973]

As stated above, the three-dimensional data encoding device according to the present embodiment performs the process illustrated in FIG. 161. First, the three-dimensional data encoding device appends, to a bitstream, first information (mode information) indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2 (S2221). In other words, the three-dimensional data encoding device encodes the first information.

[0974]

When the first information indicates that the leaf is to include a single three-dimensional point (YES in S2222), the three-dimensional data encoding device generates an N-ary tree structure in which a leaf includes a single three-dimensional point (S2223), and encodes the N-ary tree structure (S2224).

[0975]

In contrast, when the first information indicates that the leaf is to include two or more three-dimensional points, the three-dimensional data encoding device generates an N-ary tree structure in which a leaf includes two or more three-dimensional points (S2225), and encodes the N-ary tree structure (S2226).

[0976]

With this, the three-dimensional data encoding device can selectively use the tree structure in which the leaf includes a single three-dimensional point, and the tree structure in which the leaf includes two or more three-dimensional points. Accordingly, the three-dimensional data encoding device can improve the coding efficiency.

[0977]

For example, when the first information indicates that the leaf is to include two or more three-dimensional points, the three-dimensional data encoding device appends second information about the leaf (leaf information). When the first information indicates that the leaf is to include a single three-dimensional point, the three-dimensional data encoding device appends no second information to the bitstream.

[0978]

With this, the three-dimensional data encoding device can improve the coding efficiency by appending no second information to the bitstream when the leaf is to include a single three-dimensional point.

[0979]

For example, the second information indicates a total number of three-dimensional points included in the leaf.

[0980]

For example, the first information is commonly used by leaves. For example, the first information is commonly used by all or part of leaves to be included in an N-ary tree structure. In other words, the first information indicates whether each of leaves to be included in the N-ary tree structure is to include a single three-dimensional point or two or more three-dimensional points. When the first information indicates that each of the leaves is to include a single three-dimensional point, the three-dimensional data encoding device generates an N-ary tree structure in which each of leaves includes a single three-dimensional point, and encodes the N-ary tree structure. When the first information indicates that each of the leaves is to include two or more three-dimensional points, the three-dimensional data

encoding device generates an N-ary tree structure in which each of leaves includes two or more three-dimensional points, and encodes the N-ary tree structure.

[0981]

With this, since it is possible to control the formats of the leaves using the single first information, the three-dimensional data encoding device can improve the coding efficiency.

[0982]

For example, the two or more three-dimensional points included in the leaf have mutually different space coordinates. In other words, the second information may show the coordinates of each of the two or more three-dimensional points included in the leaf.

[0983]

For example, the two or more three-dimensional points included in the leaf have same space coordinates and mutually different attribute information. In other words, the second information may show the attribute information of each of the two or more three-dimensional points included in the leaf.

[0984]

For example, each of the two or more three-dimensional points included in the leaf has coordinate information and attribute information. In other words, the second information may show the coordinates and the attribute information of each of the two or more three-dimensional points included in the leaf.

[0985]

For example, the three-dimensional data encoding device includes a processor and memory, and the processor performs the above process using the memory.

[0986]

The three-dimensional data decoding device according to the present embodiment performs the process illustrated in FIG. 162. First, the three-dimensional data decoding device decodes, from a bitstream, first information (mode information) indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in

three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2 (S2231). In other words, the three-dimensional data decoding device obtains the first information from the bitstream.

[0987]

When the first information indicates that the leaf is to include a single three-dimensional point (YES in S2232), the three-dimensional data decoding device decodes an N-ary tree structure in which a leaf includes a single three-dimensional point (S2233). In contrast, when the first information indicates that the leaf is to include two or more three-dimensional points (NO in S2232), the three-dimensional data decoding device decodes an N-ary tree structure in which a leaf includes two or more three-dimensional points (S2234). Here, the phrase "to decode an N-ary tree structure in which a leaf includes a single three-dimensional point" means to parse information included in a bitstream based on a rule created on, for example, a premise that a leaf is to include only a single three-dimensional point. Likewise, the phrase "to decode an N-ary tree structure in which a leaf includes two or more three-dimensional points" means to parse information included in a bitstream based on a rule created on, for example, a premise that a leaf is to include two or more three-dimensional points.

[0988]

With this, the three-dimensional data decoding device can selectively use the tree structure in which the leaf includes a single three-dimensional point, and the tree structure in which the leaf includes two or more three-dimensional points. Accordingly, the three-dimensional data decoding device can improve the coding efficiency.

[0989]

For example, when the first information indicates that the leaf is to include two or more three-dimensional points, the three-dimensional data decoding device decodes (obtains), from the bitstream, second information about the leaf (leaf information). When the first

information indicates that the leaf is to include a single three-dimensional point, the three-dimensional data decoding device decodes (obtains) no leaf information.

[0990]

With this, since the second information need not be appended to the bitstream when the leaf is to include a single three-dimensional point, it is possible to improve the coding efficiency.

[0991]

For example, the second information indicates a total number of three-dimensional points included in the leaf.

[0992]

For example, the first information is commonly used by leaves. For example, the first information is commonly used by all or part of leaves to be included in an N-ary tree structure. In other words, the first information indicates whether each of leaves to be included in the N-ary tree structure is to include a single three-dimensional point or two or more three-dimensional points. When the first information indicates that each of the leaves is to include a single three-dimensional point, the three-dimensional data decoding device generates an N-ary tree structure in which each of leaves includes a single three-dimensional point, and decodes the N-ary tree structure. When the first information indicates that each of the leaves is to include two or more three-dimensional points, the three-dimensional data decoding device generates an N-ary tree structure in which each of leaves includes two or more three-dimensional points, and decodes the N-ary tree structure.

[0993]

With this, since it is possible to control the formats of the leaves using the single first information, the three-dimensional data decoding device can improve the coding efficiency.

[0994]

For example, the two or more three-dimensional points included in the leaf have mutually different space coordinates. In other words, the second information may show the

coordinates of each of the two or more three-dimensional points included in the leaf.

[0995]

For example, the two or more three-dimensional points included in the leaf have same space coordinates and mutually different attribute information. In other words, the second information may show the attribute information of each of the two or more three-dimensional points included in the leaf.

[0996]

For example, each of the two or more three-dimensional points included in the leaf has coordinate information and attribute information. In other words, the second information may show the coordinates and the attribute information of each of the two or more three-dimensional points included in the leaf.

[0997]

For example, the three-dimensional data decoding device includes a processor and memory, and the processor performs the above process using the memory.

[0998]

A three-dimensional data encoding device, a three-dimensional data decoding device, and the like according to the embodiments of the present disclosure have been described above, but the present disclosure is not limited to these embodiments.

[0999]

Note that each of the processors included in the three-dimensional data encoding device, the three-dimensional data decoding device, and the like according to the above embodiments is typically implemented as a large-scale integrated (LSI) circuit, which is an integrated circuit (IC). These may take the form of individual chips, or may be partially or entirely packaged into a single chip.

[1000]

Such IC is not limited to an LSI, and thus may be implemented as a dedicated circuit or a general-purpose processor. Alternatively, a field programmable gate array (FPGA) that

allows for programming after the manufacture of an LSI, or a reconfigurable processor that allows for reconfiguration of the connection and the setting of circuit cells inside an LSI may be employed.

[1001]

Moreover, in the above embodiments, the structural components may be implemented as dedicated hardware or may be realized by executing a software program suited to such structural components. Alternatively, the structural components may be implemented by a program executor such as a CPU or a processor reading out and executing the software program recorded in a recording medium such as a hard disk or a semiconductor memory.

[1002]

The present disclosure may also be implemented as a three-dimensional data encoding method, a three-dimensional data decoding method, or the like executed by the three-dimensional data encoding device, the three-dimensional data decoding device, and the like.

[1003]

Also, the divisions of the functional blocks shown in the block diagrams are mere examples, and thus a plurality of functional blocks may be implemented as a single functional block, or a single functional block may be divided into a plurality of functional blocks, or one or more functions may be moved to another functional block. Also, the functions of a plurality of functional blocks having similar functions may be processed by single hardware or software in a parallelized or time-divided manner.

[1004]

Also, the processing order of executing the steps shown in the flowcharts is a mere illustration for specifically describing the present disclosure, and thus may be an order other than the shown order. Also, one or more of the steps may be executed simultaneously (in parallel) with another step.

[1005]

A three-dimensional data encoding device, a three-dimensional data decoding device, and the like according to one or more aspects have been described above based on the embodiments, but the present disclosure is not limited to these embodiments. The one or more aspects may thus include forms achieved by making various modifications to the above embodiments that can be conceived by those skilled in the art, as well forms achieved by combining structural components in different embodiments, without materially departing from the spirit of the present disclosure.

INDUSTRIAL APPLICABILITY

[1006]

The present disclosure is applicable to a three-dimensional data encoding device and a three-dimensional data decoding device.

REFERENCE MARKS IN THE DRAWINGS

[1007]

100, 400 three-dimensional data encoding device

101, 201, 401, 501 obtainer

102, 402 encoding region determiner

103 divider

104, 644 encoder

111 three-dimensional data

112, 211, 413, 414, 511, 634 encoded three-dimensional data

200, 500 three-dimensional data decoding device

202 decoding start GOS determiner

203 decoding SPC determiner

204, 625 decoder

212, 512, 513 decoded three-dimensional data

403 SWLD extractor

404 WLD encoder

405 SWLD encoder

411 input three-dimensional data

412 extracted three-dimensional data

502 header analyzer

503 WLD decoder

504 SWLD decoder

620, 620A three-dimensional data creation device

621, 641 three-dimensional data creator

622 request range determiner

623 searcher

624, 642 receiver

626 merger

631, 651 sensor information

632 first three-dimensional data

633 request range information

635 second three-dimensional data

636 third three-dimensional data

640 three-dimensional data transmission device

643 extractor

645 transmitter

652 fifth three-dimensional data

654 sixth three-dimensional data

700 three-dimensional information processing device

701 three-dimensional map obtainer

702 self-detected data obtainer

703 abnormal case judgment unit

704 coping operation determiner

705 operation controller

711 three-dimensional map

712 self-detected three-dimensional data

810 three-dimensional data creation device

811 data receiver

812, 819 communication unit

813 reception controller

814, 821 format converter

815 sensor

816 three-dimensional data creator

817 three-dimensional data synthesizer

818 three-dimensional data storage

820 transmission controller

822 data transmitter

831, 832, 834, 835, 836, 837 three-dimensional data

833 sensor information

901 server

902, 902A, 902B, 902C client device

1011, 1111 data receiver

1012, 1020, 1112, 1120 communication unit

1013, 1113 reception controller

1014, 1019, 1114, 1119 format converter

1015 sensor

1016, 1116 three-dimensional data creator

1017 three-dimensional image processor

1018, 1118 three-dimensional data storage

1021, 1121 transmission controller

1022, 1122 data transmitter

1031, 1032, 1135 three-dimensional map

1033, 1037, 1132 sensor information

1034, 1035, 1134 three-dimensional data

1117 three-dimensional data merger

1201 three-dimensional map compression/decoding processor

1202 sensor information compression/decoding processor

1211 three-dimensional map decoding processor

1212 sensor information compression processor

1300 three-dimensional data encoding device

1301 divider

1302 subtractor

1303 transformer

1304 quantizer

1305, 1402 inverse quantizer

1306, 1403 inverse transformer

1307, 1404 adder

1308, 1405 reference volume memory

1309, 1406 intra predictor

1310, 1407 reference space memory

1311, 1408 inter predictor

1312, 1409 prediction controller

1313 entropy encoder

1400 three-dimensional data decoding device

1401 entropy decoder

1501 server

1502 client

1511 storage

1512 controller

1513 encoded three-dimensional map

1521 decoder

1522 application

1900 three-dimensional data encoding device

1901, 1911 octree generator

1902, 1912 similarity information calculator

1903, 1913 coding table selector

1904 entropy encoder

1910 three-dimensional data decoding device

1914 entropy decoder

2100 three-dimensional data encoding device

2101, 2111 octree generator

2102, 2112 geometry information calculator

2103, 2113 coding table selector

2104 entropy encoder

2110 three-dimensional data decoding device

2114 entropy decoder

2200 three-dimensional data encoding device

2201, 2211 octree generator

2202 mode determiner

2203 entropy encoder

2210 three-dimensional data decoding device

2212 mode information decoder

2213 entropy decoder

CLAIMS

1. A three-dimensional data encoding method, comprising:

appending, to a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2;

when the first information indicates that the leaf is to include a single three-dimensional point, generating an N-ary tree structure in which a leaf includes a single three-dimensional point, and encoding the N-ary tree structure; and

when the first information indicates that the leaf is to include two or more three-dimensional points, generating an N-ary tree structure in which a leaf includes two or more three-dimensional points, and encoding the N-ary tree structure.

2. The three-dimensional data encoding method according to claim 1, further comprising:

when the first information indicates that the leaf is to include two or more three-dimensional points, appending second information about the leaf to the bitstream,

wherein when the first information indicates that the leaf is to include a single three-dimensional point, no second information is appended to the bitstream.

3. The three-dimensional data encoding method according to claim 2,

wherein the second information indicates a total number of three-dimensional points included in the leaf.

4. The three-dimensional data encoding method according to any one of claims 1 to 3,

wherein the first information indicates whether each of leaves to be included in the N-ary tree structure is to include a single three-dimensional point or two or more three-dimensional points,

when the first information indicates that each of the leaves is to include a single three-dimensional point, an N-ary tree structure is generated in which each of leaves includes a single three-dimensional point, and the N-ary tree structure is encoded, and

when the first information indicates that each of the leaves is to include two or more three-dimensional points, an N-ary tree structure is generated in which each of leaves includes two or more three-dimensional points, and the N-ary tree structure is encoded.

5. The three-dimensional data encoding method according to any one of claims 1 to 4,

wherein the two or more three-dimensional points included in the leaf have mutually different space coordinates.

6. The three-dimensional data encoding method according to any one of claims 1 to 4,

wherein the two or more three-dimensional points included in the leaf have same space coordinates and mutually different attribute information.

7. The three-dimensional data encoding method according to any one of claims 1 to 4,

wherein each of the two or more three-dimensional points included in the leaf has coordinate information and attribute information.

8. A three-dimensional data decoding method, comprising:

decoding, from a bitstream, first information indicating whether a leaf to be included

in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2;

when the first information indicates that the leaf is to include a single three-dimensional point, decoding an N-ary tree structure in which a leaf includes a single three-dimensional point; and

when the first information indicates that the leaf is to include two or more three-dimensional points, decoding an N-ary tree structure in which a leaf includes two or more three-dimensional points.

9. The three-dimensional data decoding method according to claim 8, further comprising:

when the first information indicates that the leaf is to include two or more three-dimensional points, decoding, from the bitstream, second information about the leaf,

wherein when the first information indicates that the leaf is to include a single three-dimensional point, no second information is decoded from the bitstream.

10. The three-dimensional data decoding method according to claim 9,

wherein the second information indicates a total number of three-dimensional points included in the leaf.

11. The three-dimensional data decoding method according to any one of claims 8 to 10,

wherein the first information indicates whether each of leaves to be included in the N-ary tree structure is to include a single three-dimensional point or two or more three-dimensional points,

when the first information indicates that each of the leaves is to include a single

three-dimensional point, an N-ary tree structure is decoded in which each of leaves includes a single three-dimensional point, and

when the first information indicates that each of the leaves is to include two or more three-dimensional points, an N-ary tree structure is decoded in which each of leaves includes two or more three-dimensional points.

12. The three-dimensional data decoding method according to any one of claims 8 to 11,

wherein the two or more three-dimensional points included in the leaf have mutually different space coordinates.

13. The three-dimensional data decoding method according to any one of claims 8 to 11,

wherein the two or more three-dimensional points included in the leaf have same space coordinates and mutually different attribute information.

14. The three-dimensional data decoding method according to any one of claims 8 to 11,

wherein each of the two or more three-dimensional points included in the leaf has coordinate information and attribute information.

15. A three-dimensional data encoding device, comprising:

a processor; and

memory,

wherein using the memory, the processor:

appends, to a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional

data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2;

when the first information indicates that the leaf is to include a single three-dimensional point, generates an N-ary tree structure in which a leaf includes a single three-dimensional point, and encodes the N-ary tree structure; and

when the first information indicates that the leaf is to include two or more three-dimensional points, generates an N-ary tree structure in which a leaf includes two or more three-dimensional points, and encodes the N-ary tree structure.

16. A three-dimensional data decoding device, comprising:

a processor; and

memory,

wherein using the memory, the processor:

decodes, from a bitstream, first information indicating whether a leaf to be included in an N-ary tree structure of three-dimensional points included in three-dimensional data is to include a single three-dimensional point or two or more three-dimensional points, where N is an integer greater than or equal to 2;

when the first information indicates that the leaf is to include a single three-dimensional point, decodes an N-ary tree structure in which a leaf includes a single three-dimensional point; and

when the first information indicates that the leaf is to include two or more three-dimensional points, decodes an N-ary tree structure in which a leaf includes two or more three-dimensional points.

FIG. 1

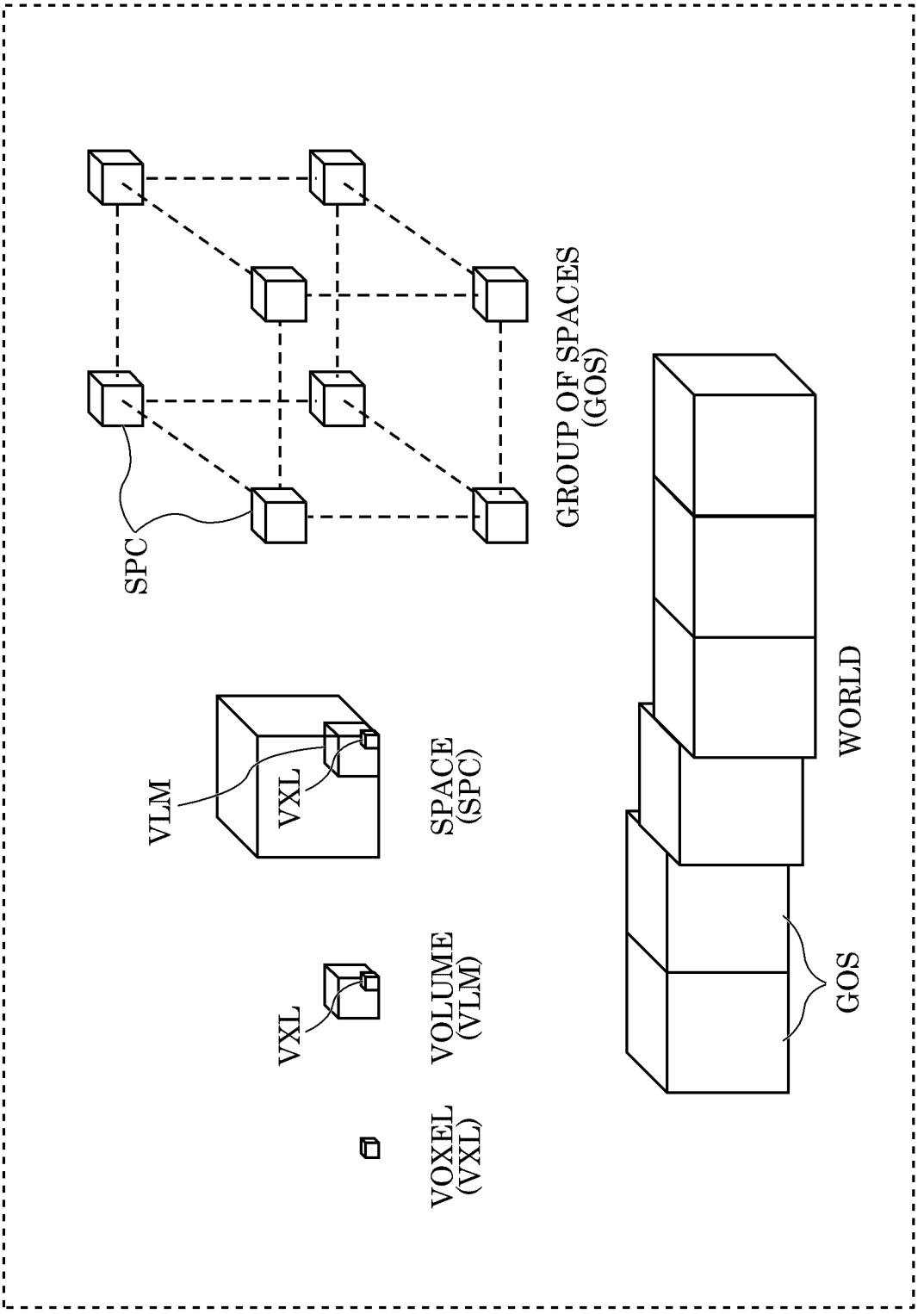

WORLD
GOS
GROUP OF SPACES (GOS)
SPC
SPACE (SPC)
VLM
VXL
VOLUME (VLM)
VXL
VOXEL (VXL)

FIG. 2

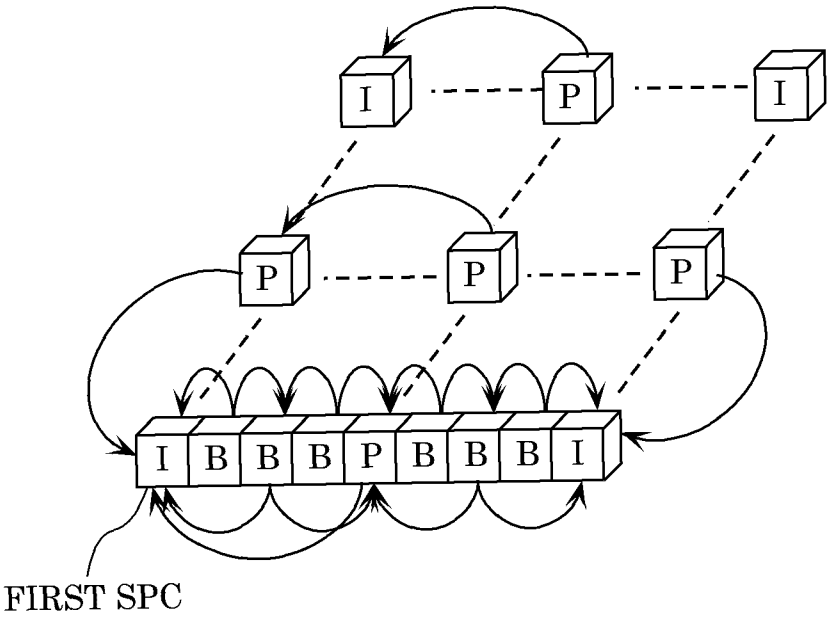


FIG. 3

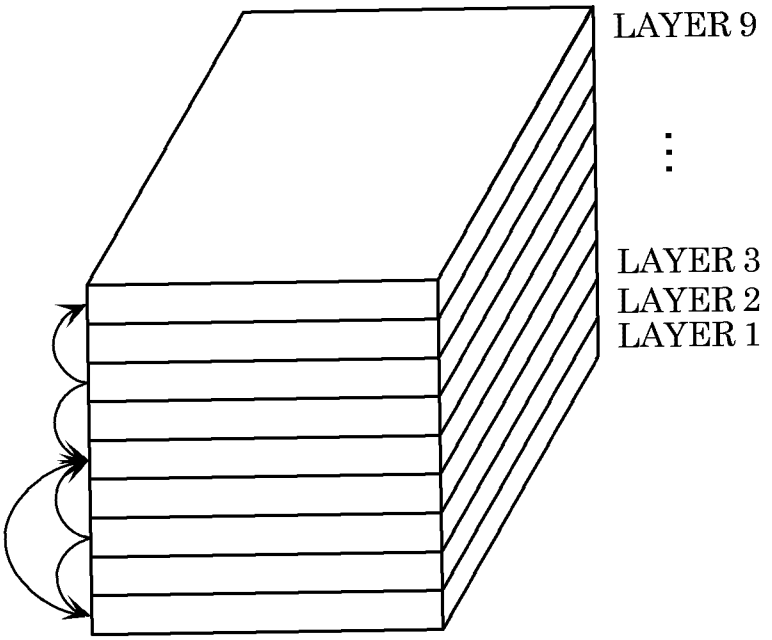


FIG. 4

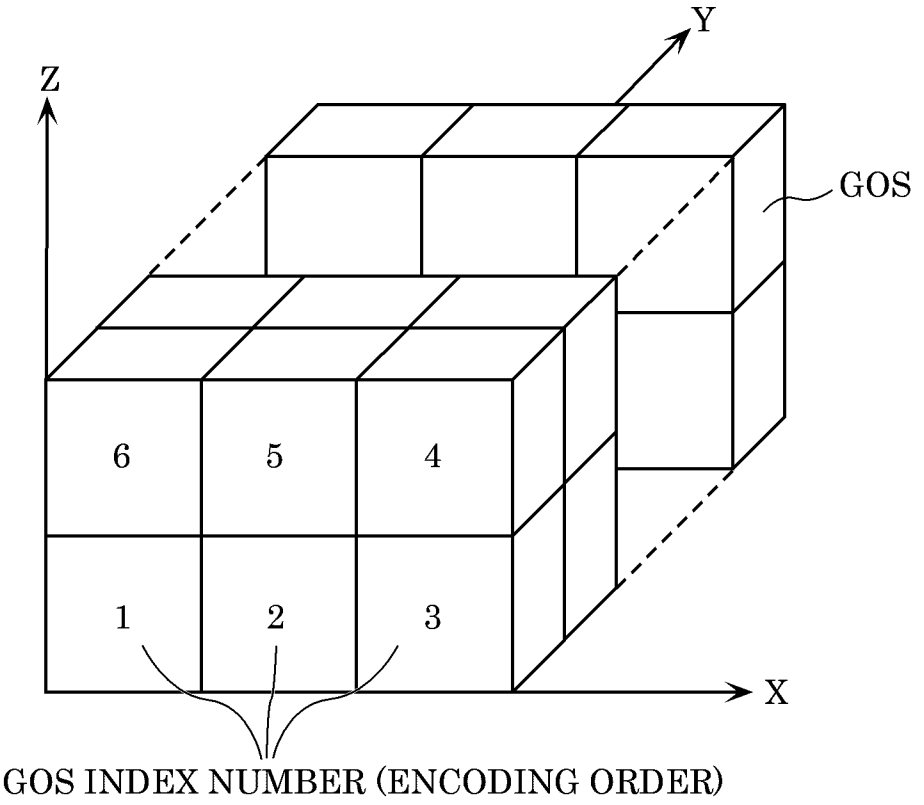

Y
Z
GOS
6
5
4
1
2
3
X
GOS INDEX NUMBER (ENCODING ORDER)

FIG. 5

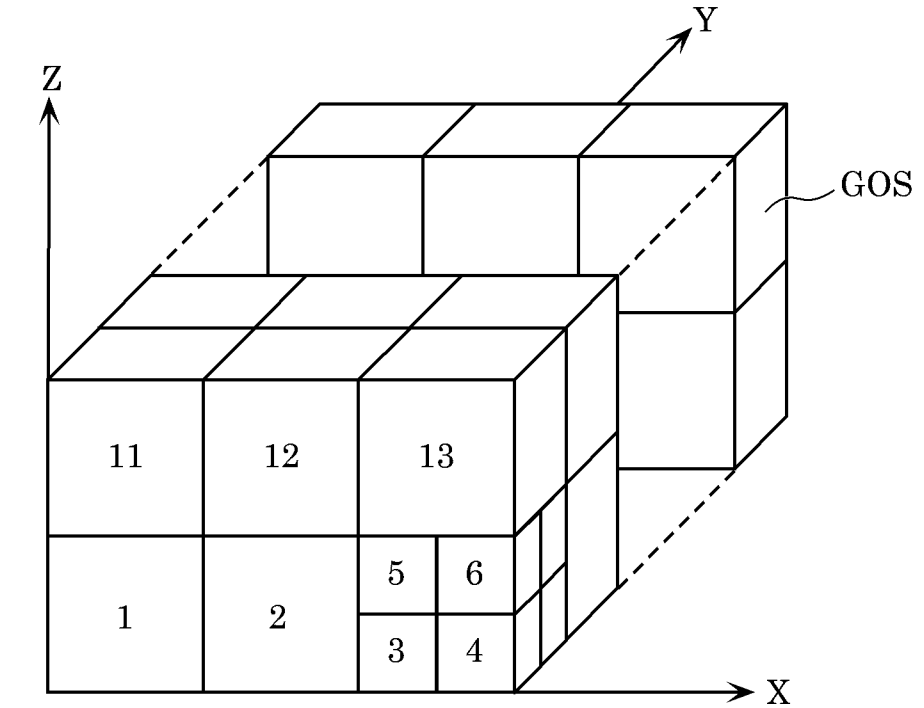

Y
Z
GOS
11
12
13
1
2
5
6
3
4
X

FIG. 6

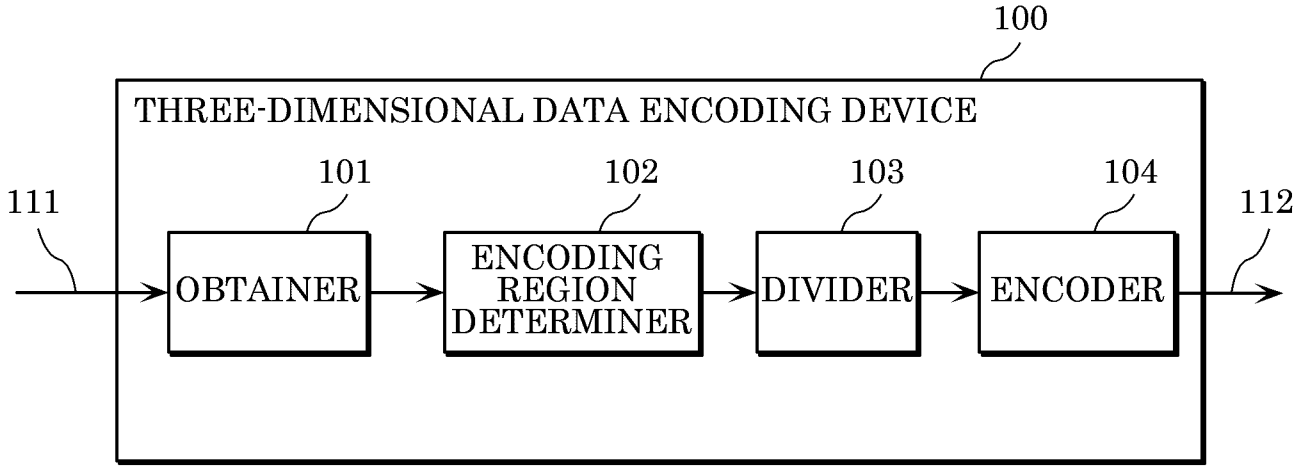


FIG. 7

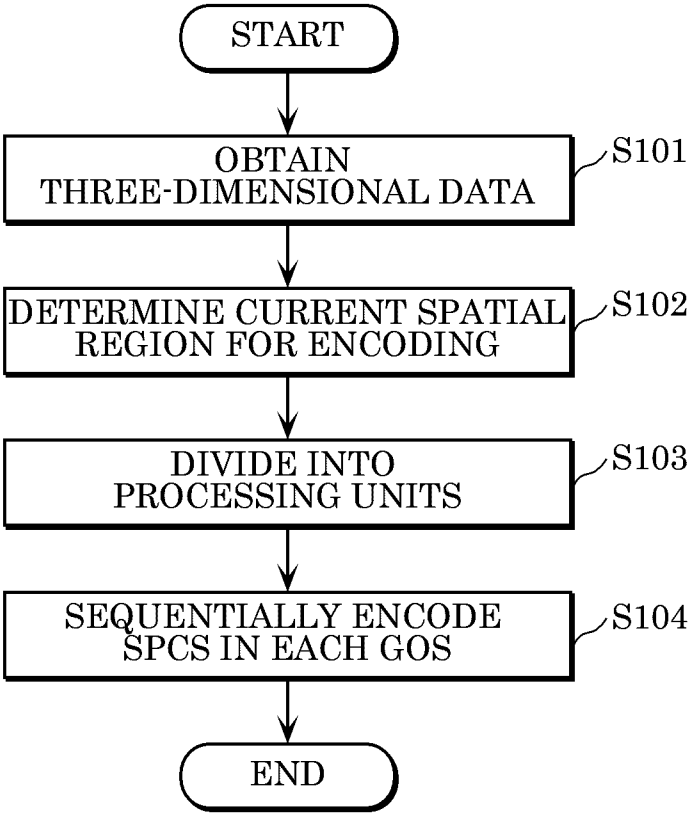


FIG. 8

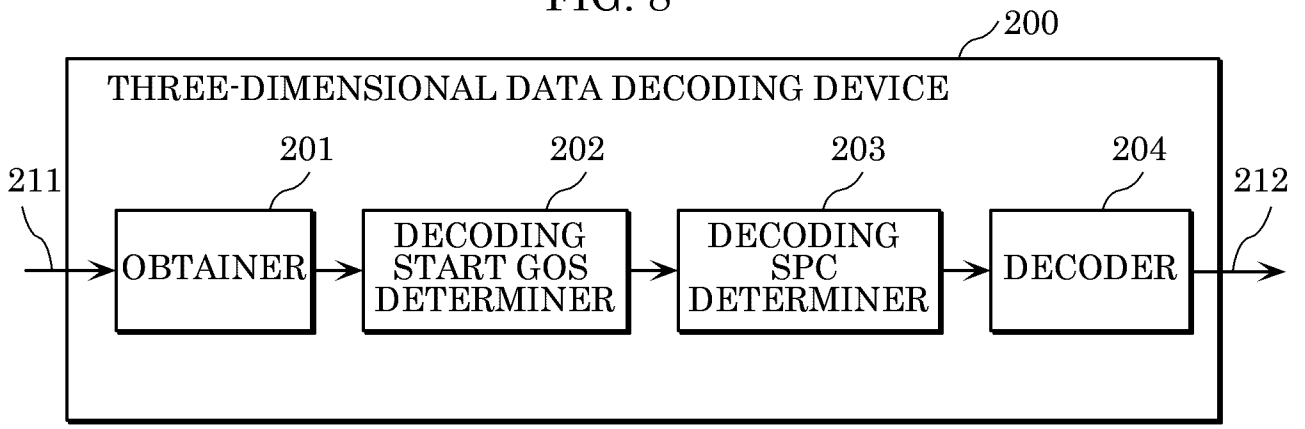


FIG. 9

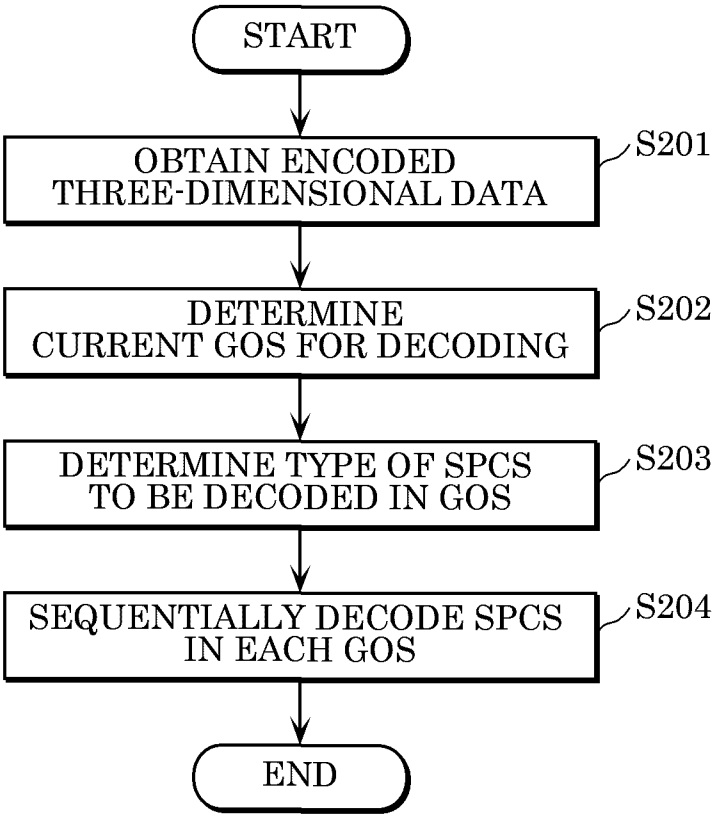


FIG. 10

OBJECT	GOS NUMBER
object(1)	1, 2
object(2)	5
object(3)	8, 9, 10
...	...
object(N)	101, 113, 125

OBJECT-GOS TABLE

COORDINATES	GOS NUMBER
(x1, y1, z1)	1
(x2, y2, z2)	2
(x3, y3, z3)	3
...	...
(xN, yN, zN)	N

COORDINATES-GOS TABLE

TIME	GOS NUMBER
pts(1)	1
pts(2)	30
pts(3)	50
...	...
pts(N)	200

TIME-GOS TABLE

GOS NUMBER	GOS ADDRESS
1	addr(1)
2	addr(2)
3	addr(3)
...	...
N	addr(N)

GOS-ADDRESS TABLE

FIG. 11

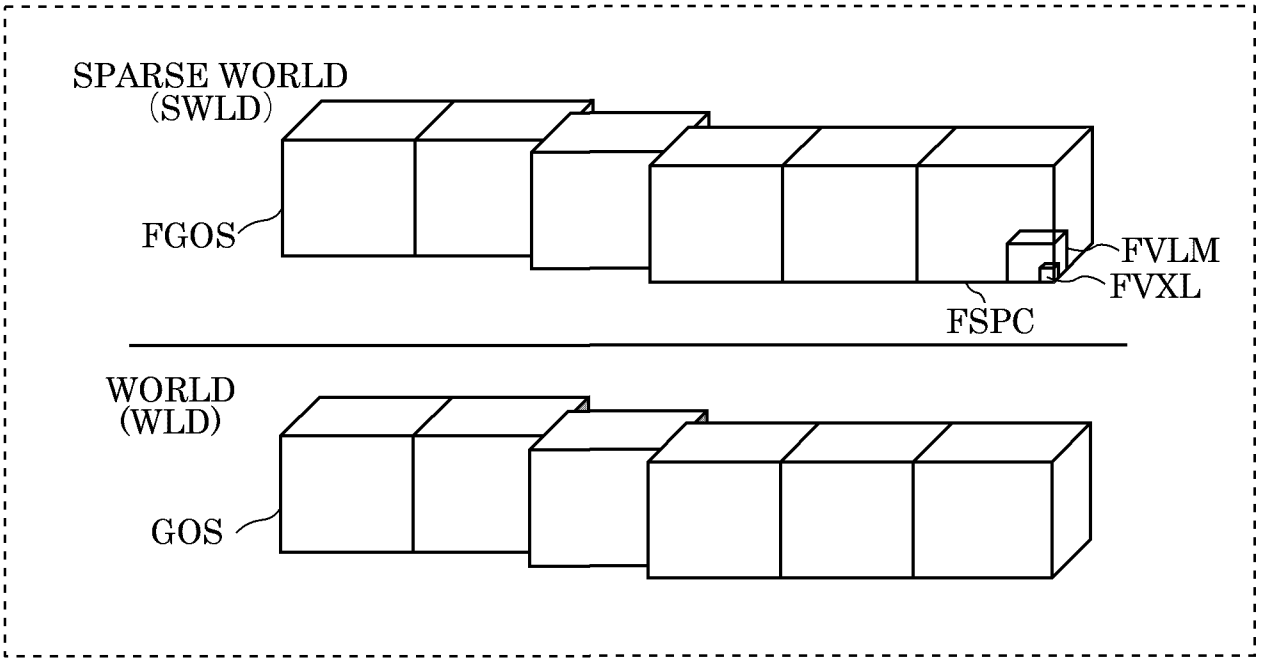

SPARSE WORLD
(SWLD)
FGOS
FVLM
FVXL
FSPC
WORLD
(WLD)
GOS

FIG. 12

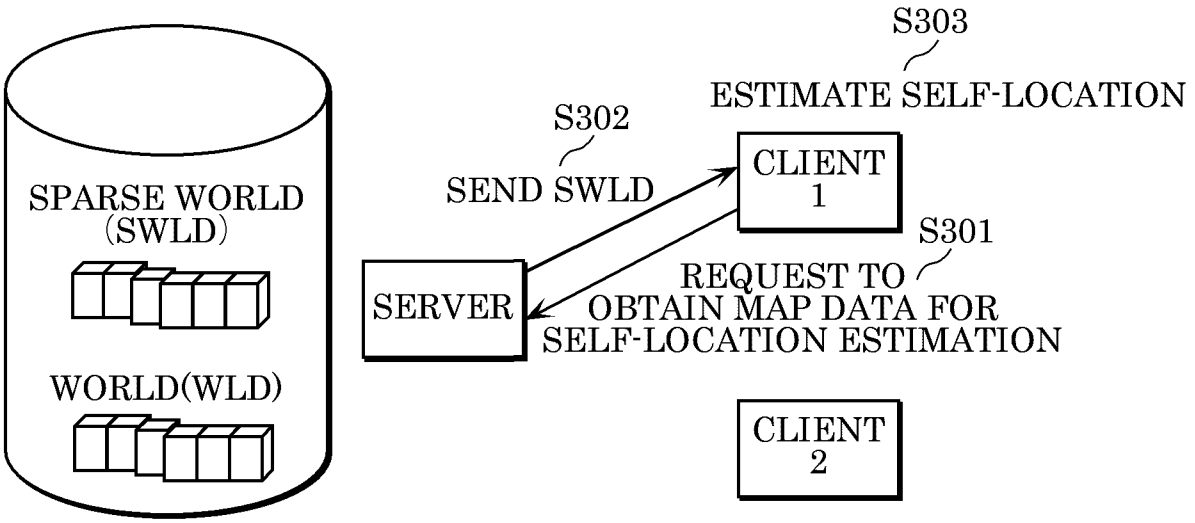

SPARSE WORLD
(SWLD)
WORLD(WLD)
S303
ESTIMATE SELF-LOCATION
S302
SEND SWLD
CLIENT
1
S301
REQUEST TO
OBTAIN MAP DATA FOR
SELF-LOCATION ESTIMATION
SERVER
CLIENT
2

FIG. 13

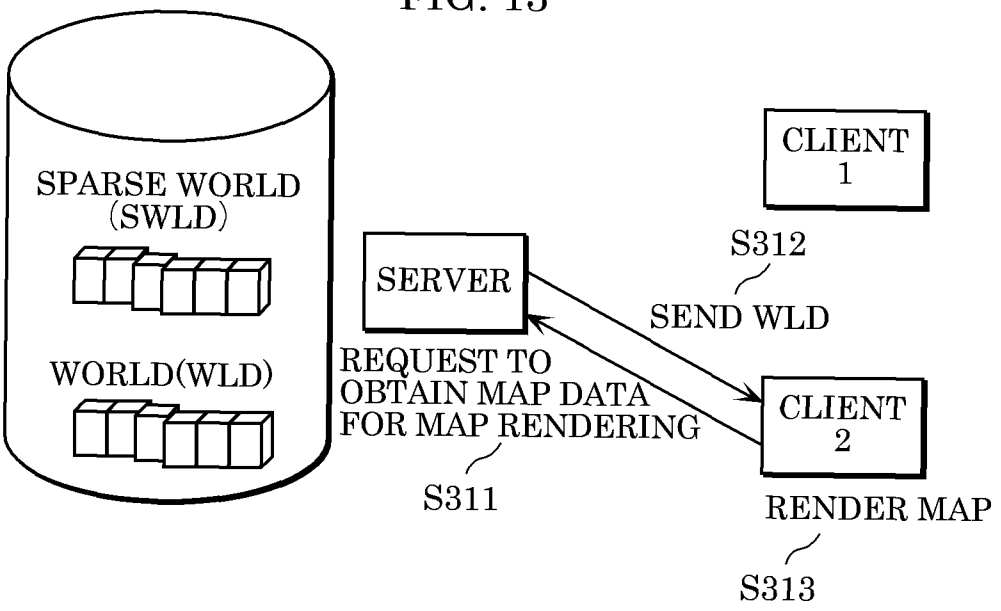


FIG. 14

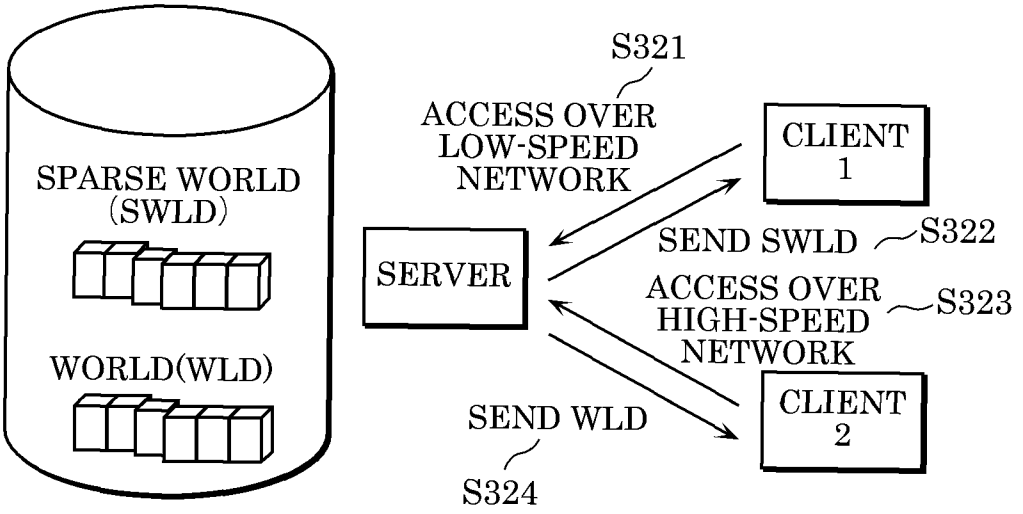


FIG. 15

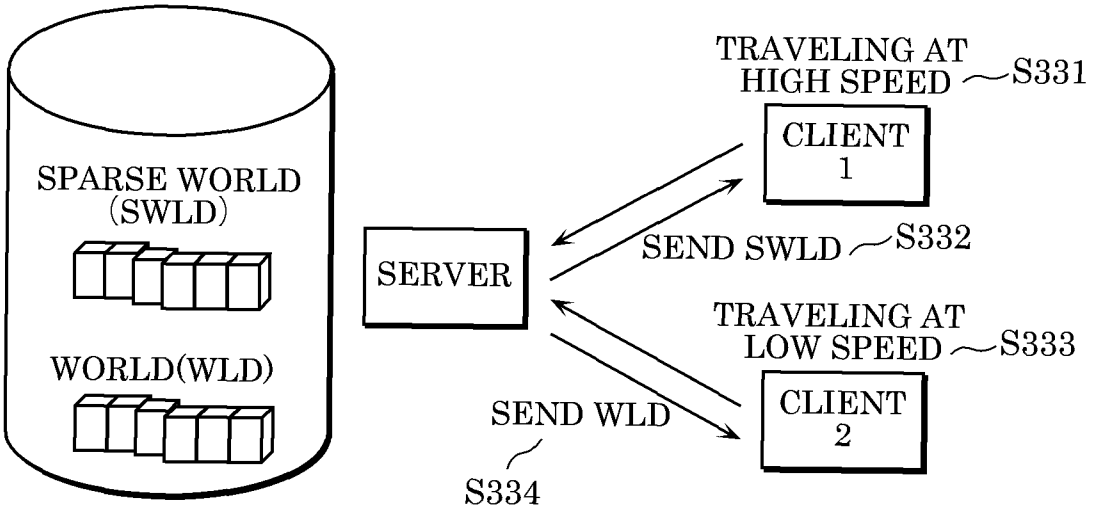


FIG. 16

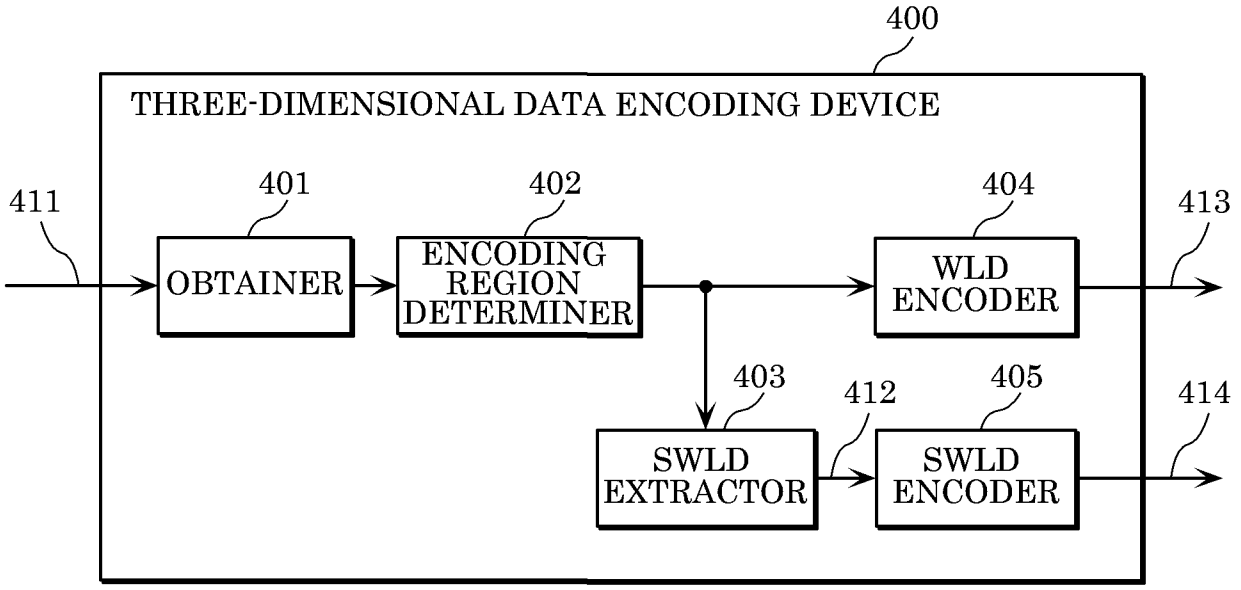

400
THREE-DIMENSIONAL DATA ENCODING DEVICE
411
401
OBTAINER
402
ENCODING REGION DETERMINER
404
WLD ENCODER
413
403
SWLD EXTRACTOR
412
405
SWLD ENCODER
414

FIG. 17

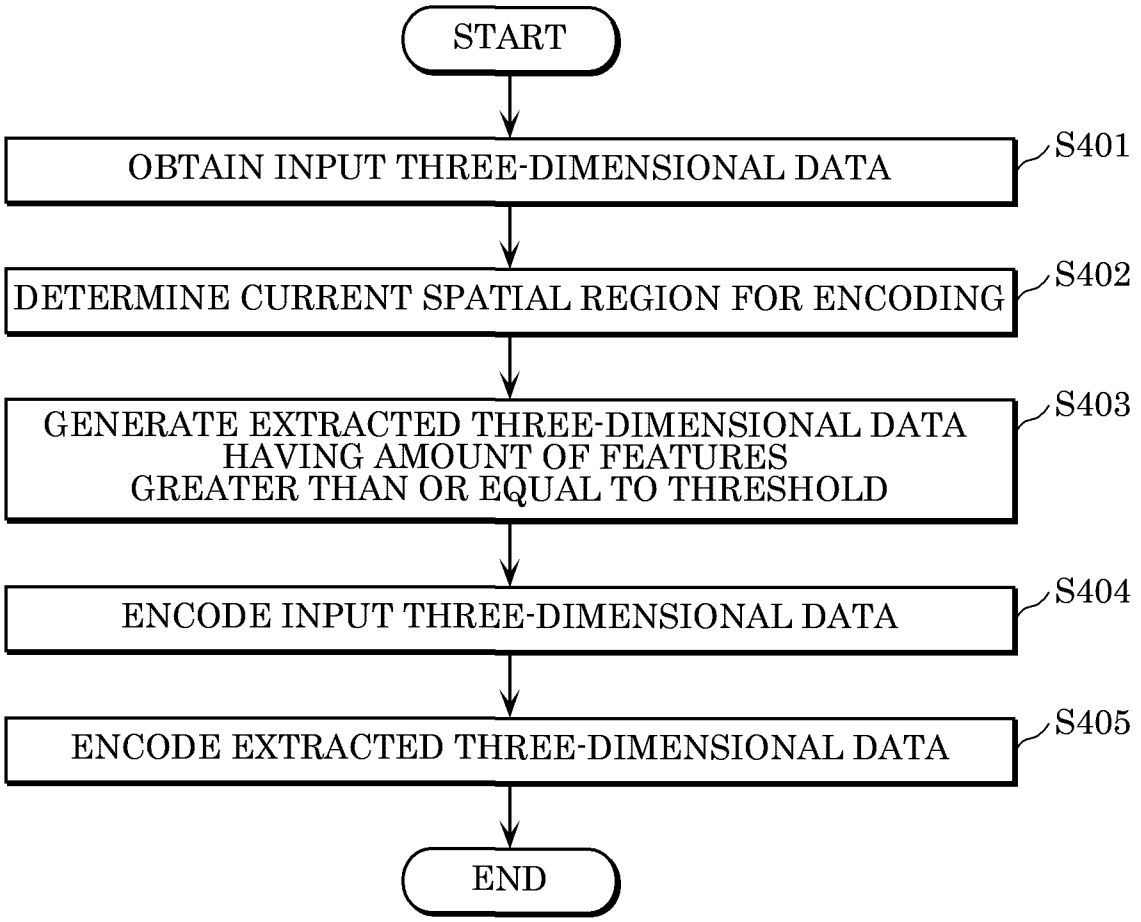

START
OBTAIN INPUT THREE-DIMENSIONAL DATA
S401
DETERMINE CURRENT SPATIAL REGION FOR ENCODING
S402
GENERATE EXTRACTED THREE-DIMENSIONAL DATA HAVING AMOUNT OF FEATURES GREATER THAN OR EQUAL TO THRESHOLD
S403
ENCODE INPUT THREE-DIMENSIONAL DATA
S404
ENCODE EXTRACTED THREE-DIMENSIONAL DATA
S405
END

FIG. 18

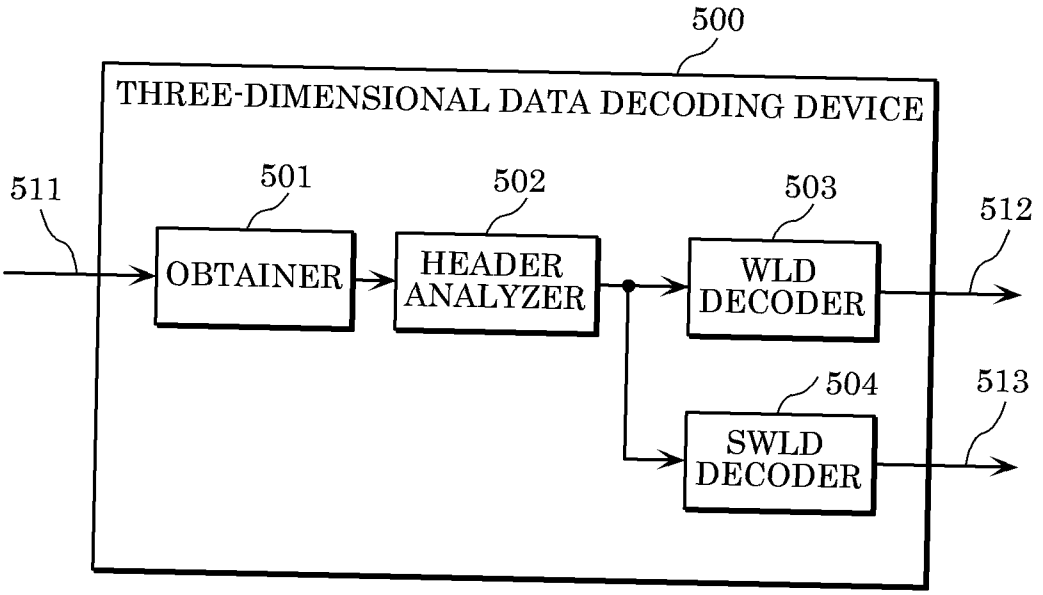

500
THREE-DIMENSIONAL DATA DECODING DEVICE
511
501
OBTAINER
502
HEADER ANALYZER
503
WLD DECODER
512
504
SWLD DECODER
513

FIG. 19

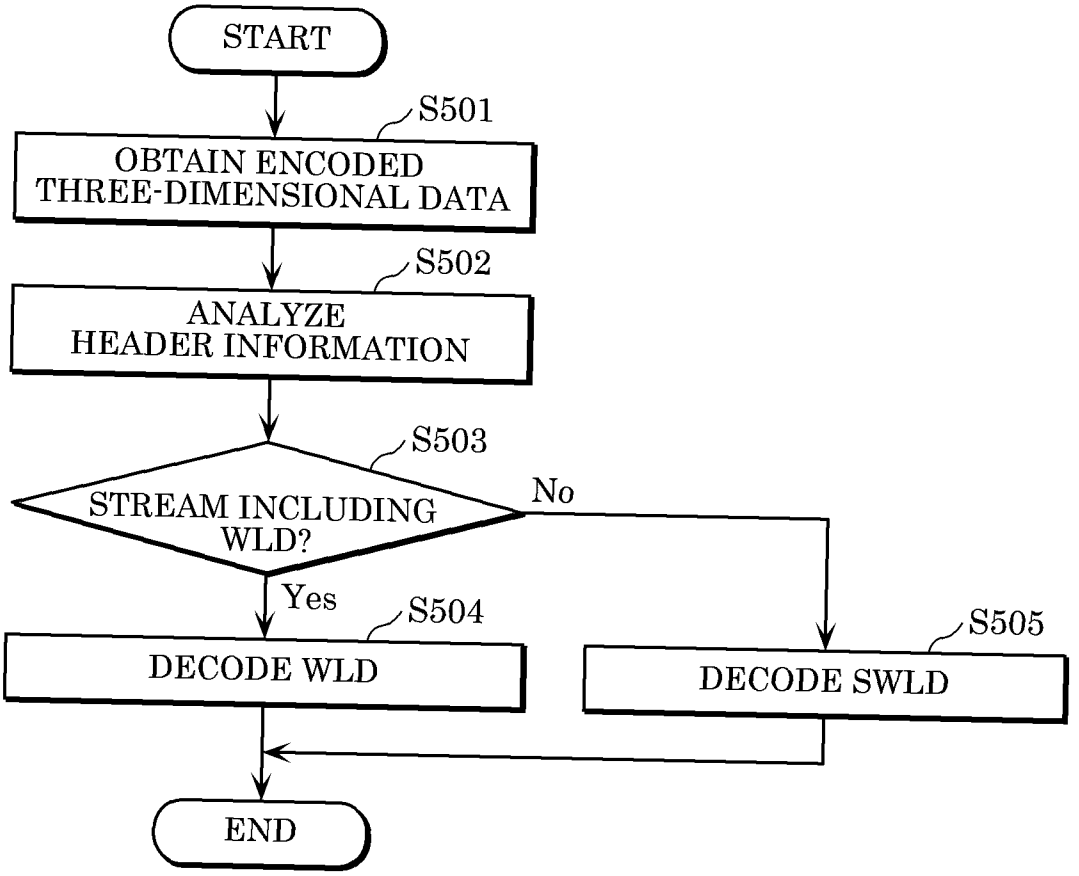

START
S501
OBTAIN ENCODED THREE-DIMENSIONAL DATA
S502
ANALYZE HEADER INFORMATION
S503
STREAM INCLUDING WLD?
No
Yes
S504
DECODE WLD
S505
DECODE SWLD
END

FIG. 20

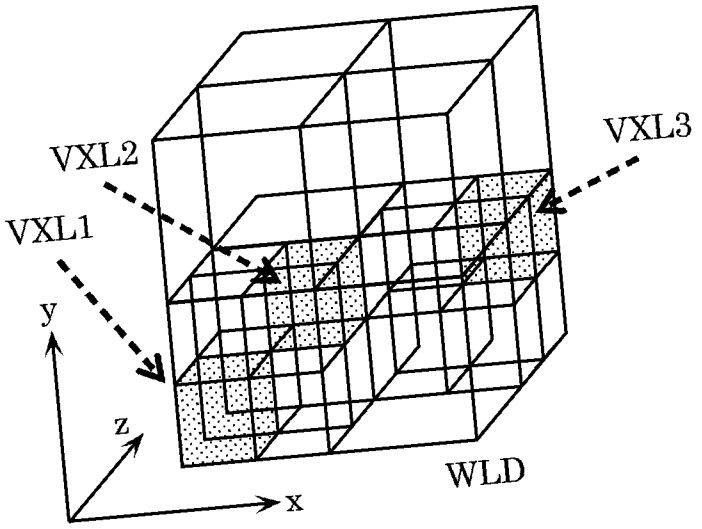


FIG. 21

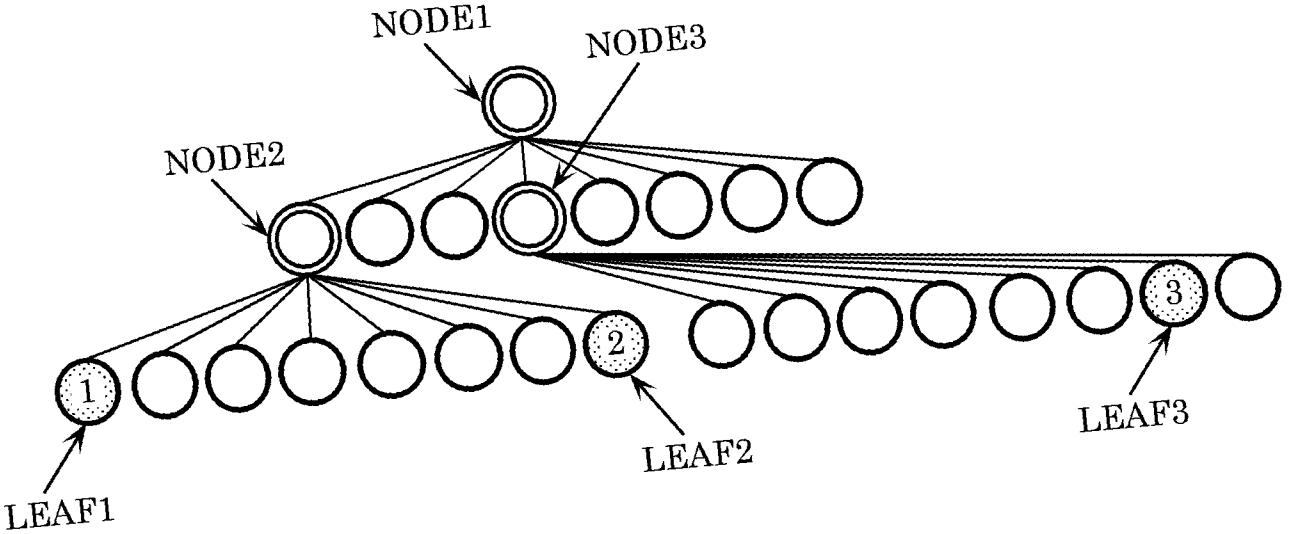


FIG. 22

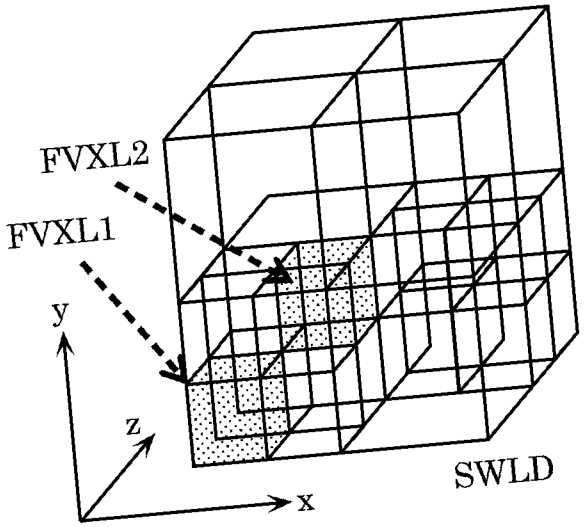


FIG. 23

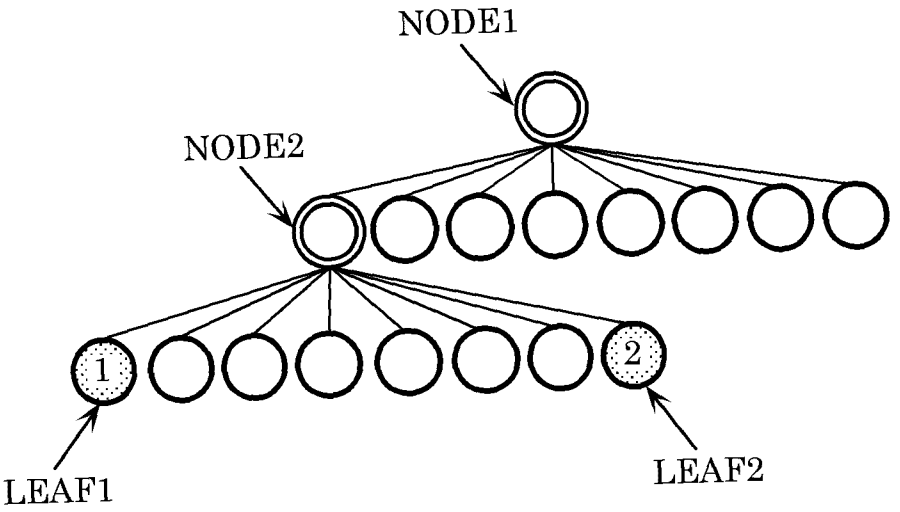

NODE1
NODE2
1
2
LEAF1
LEAF2

FIG. 24

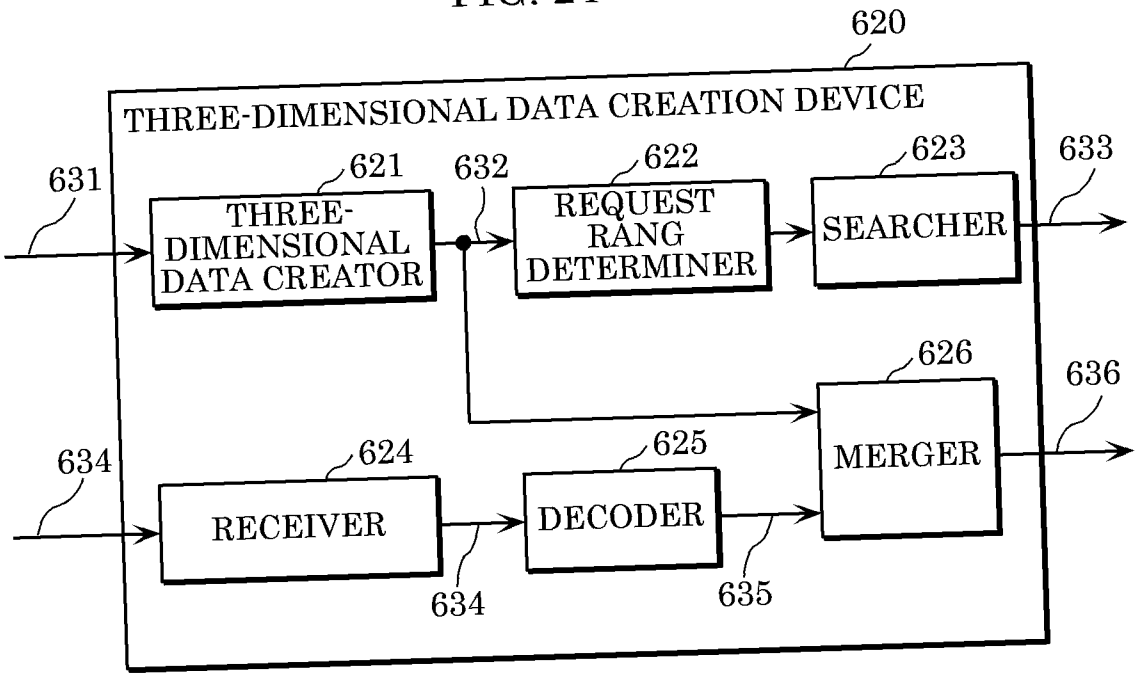

620
THREE-DIMENSIONAL DATA CREATION DEVICE
631
621
THREE-DIMENSIONAL DATA CREATOR
632
622
REQUEST RANG DETERMINER
623
SEARCHER
633
626
636
MERGER
634
624
RECEIVER
625
DECODER
634
635

FIG. 25

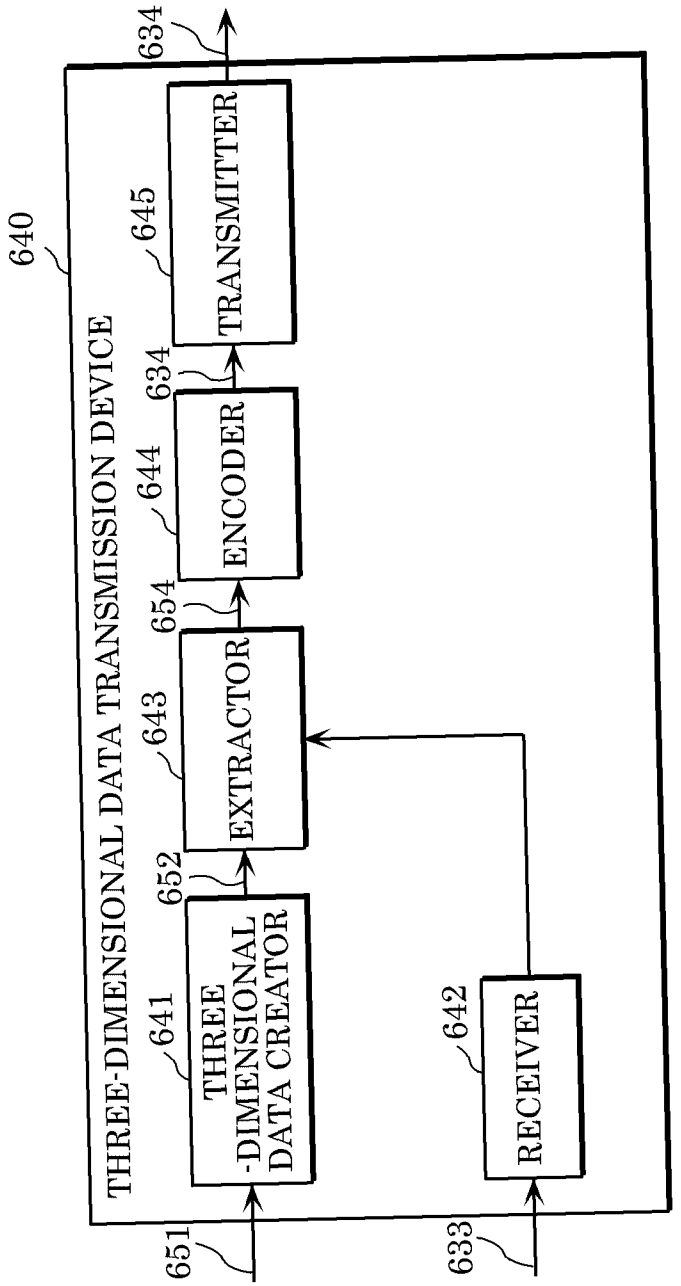

640
THREE-DIMENSIONAL DATA TRANSMISSION DEVICE
641
THREE -DIMENSIONAL DATA CREATOR
642
RECEIVER
643
EXTRACTOR
644
ENCODER
645
TRANSMITTER
651
652
654
633
634
634

FIG. 26

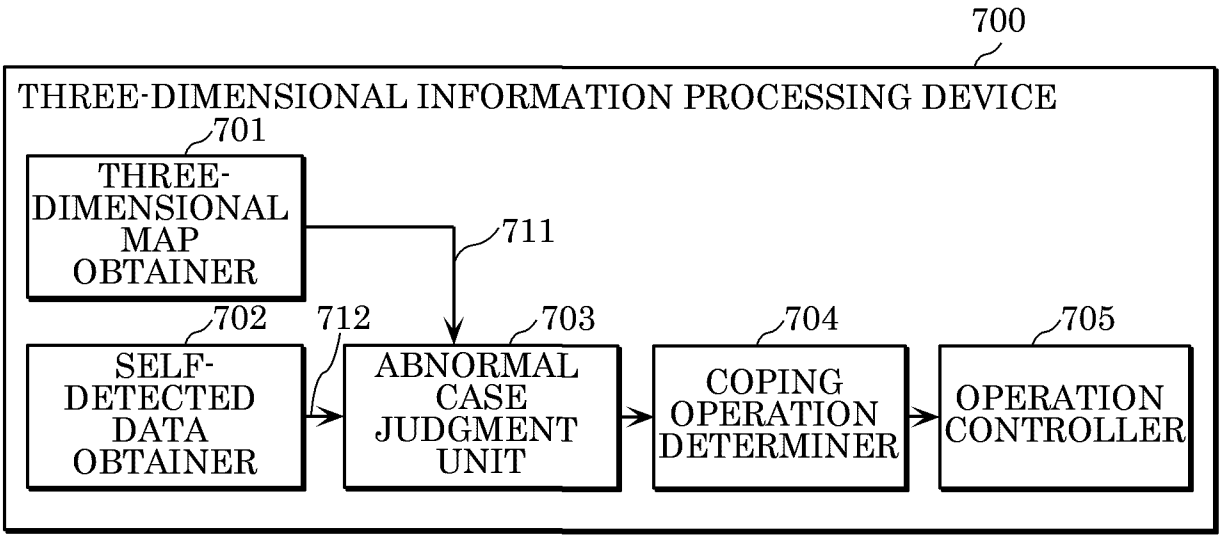

700
THREE-DIMENSIONAL INFORMATION PROCESSING DEVICE
701
THREE-DIMENSIONAL MAP OBTAINER
711
702
712
703
704
705
SELF-DETECTED DATA OBTAINER
ABNORMAL CASE JUDGMENT UNIT
COPING OPERATION DETERMINER
OPERATION CONTROLLER

FIG. 27

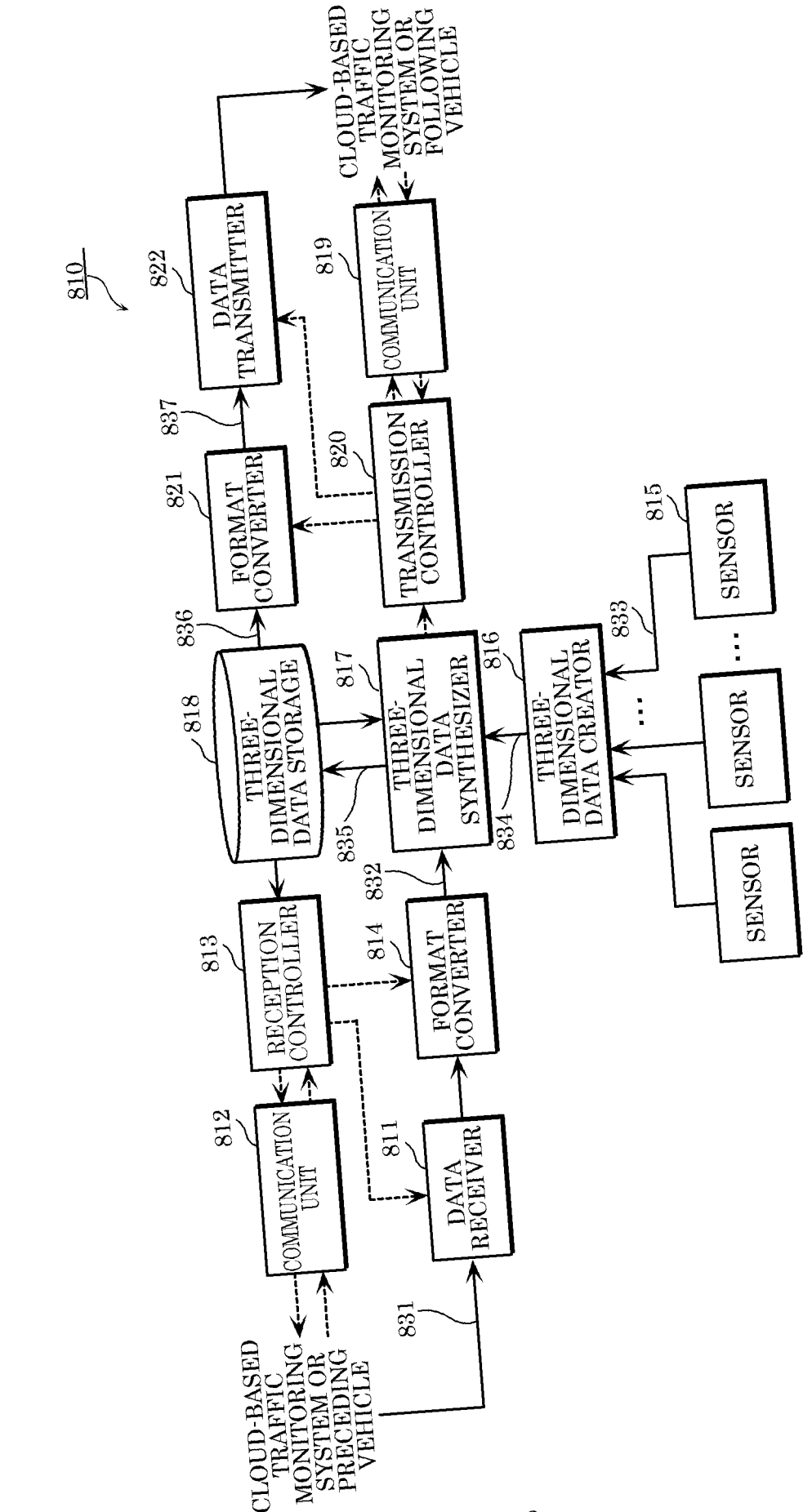

810
CLOUD-BASED TRAFFIC MONITORING SYSTEM OR PRECEDING VEHICLE
CLOUD-BASED TRAFFIC MONITORING SYSTEM OR FOLLOWING VEHICLE
COMMUNICATION UNIT
DATA RECEIVER
RECEPTION CONTROLLER
FORMAT CONVERTER
THREE-DIMENSIONAL DATA STORAGE
THREE-DIMENSIONAL DATA SYNTHESIZER
THREE-DIMENSIONAL DATA CREATOR
TRANSMISSION CONTROLLER
FORMAT CONVERTER
DATA TRANSMITTER
COMMUNICATION UNIT
SENSOR
SENSOR
SENSOR
811
812
813
814
815
816
817
818
819
820
821
822
831
832
833
834
835
836
837

FIG. 28

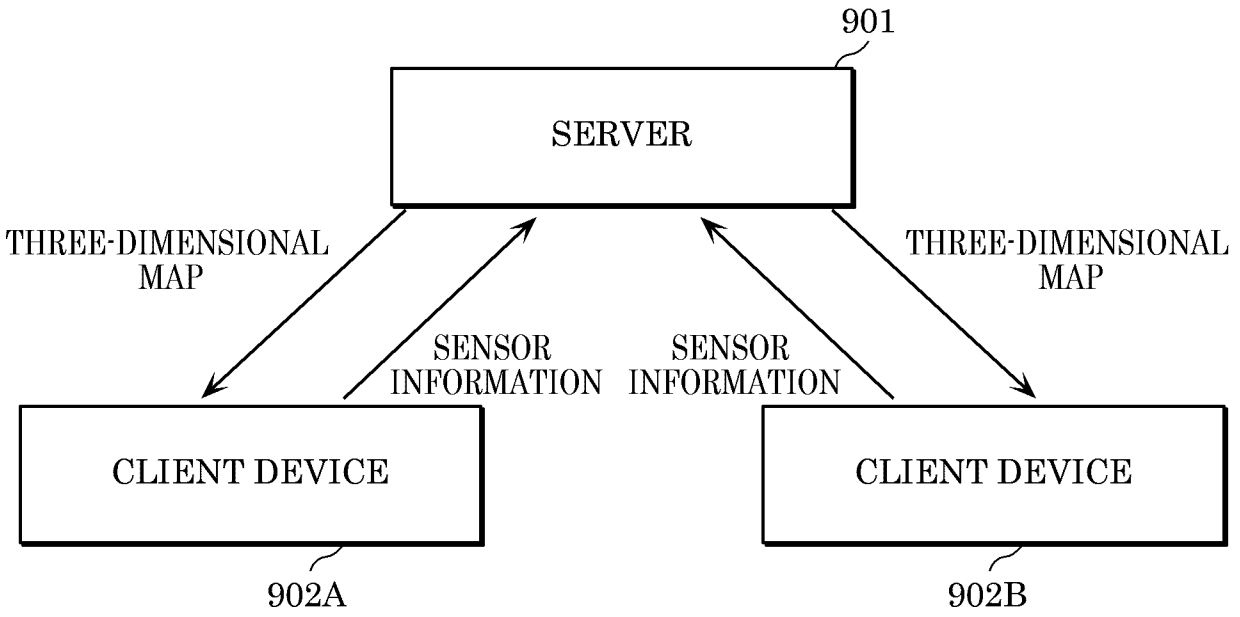

901
SERVER
THREE-DIMENSIONAL MAP
SENSOR INFORMATION
SENSOR INFORMATION
THREE-DIMENSIONAL MAP
CLIENT DEVICE
CLIENT DEVICE
902A
902B

FIG. 29

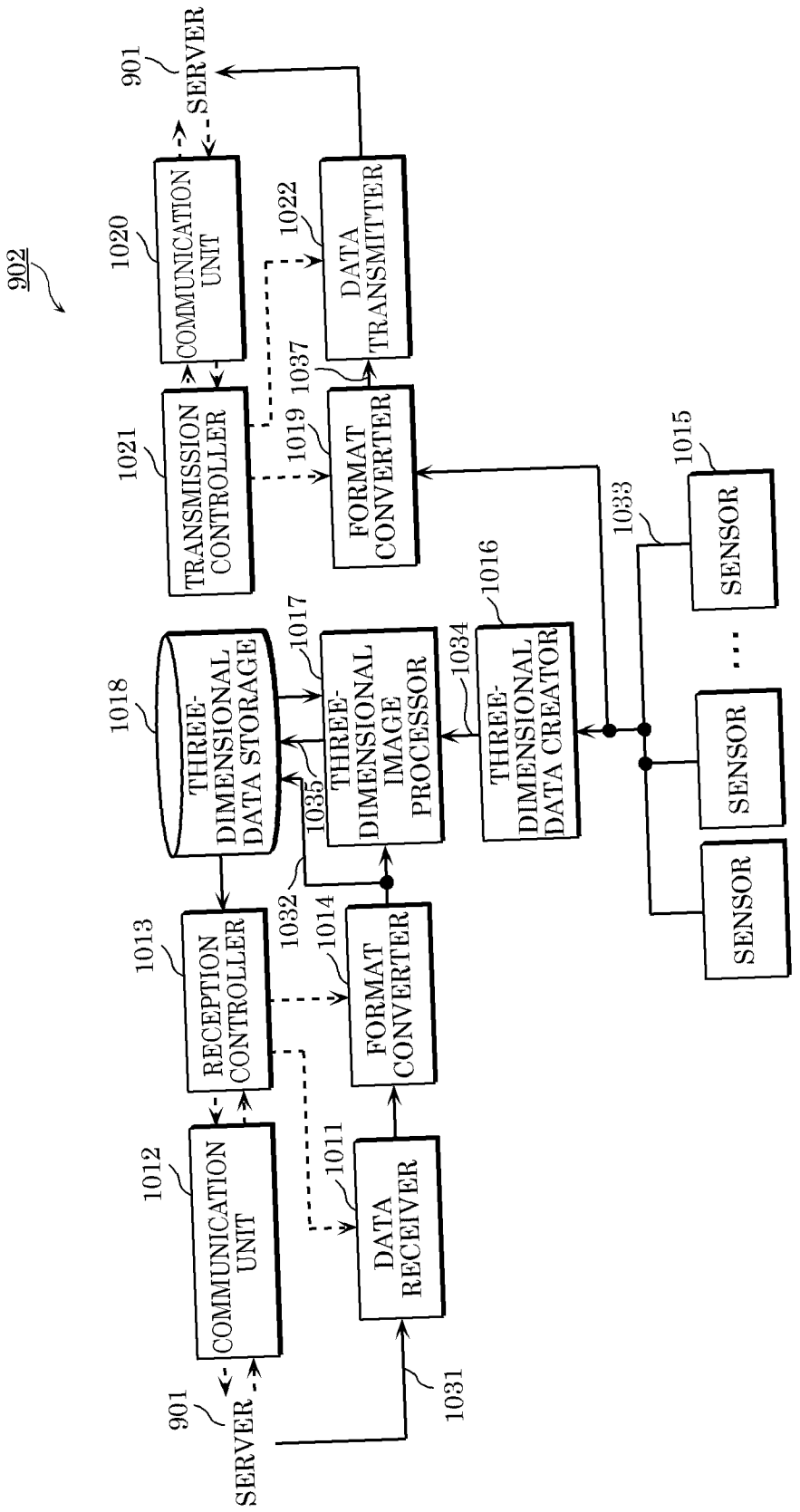

902
901
SERVER
1020
COMMUNICATION UNIT
1022
DATA TRANSMITTER
1021
TRANSMISSION CONTROLLER
1019
1037
FORMAT CONVERTER
1015
SENSOR
1033
1016
1018
THREE-DIMENSIONAL DATA STORAGE
1017
THREE-DIMENSIONAL IMAGE PROCESSOR
1034
THREE-DIMENSIONAL DATA CREATOR
1035
1013
RECEPTION CONTROLLER
1032
1014
FORMAT CONVERTER
1012
COMMUNICATION UNIT
1011
DATA RECEIVER
1031
901
SERVER

FIG. 30

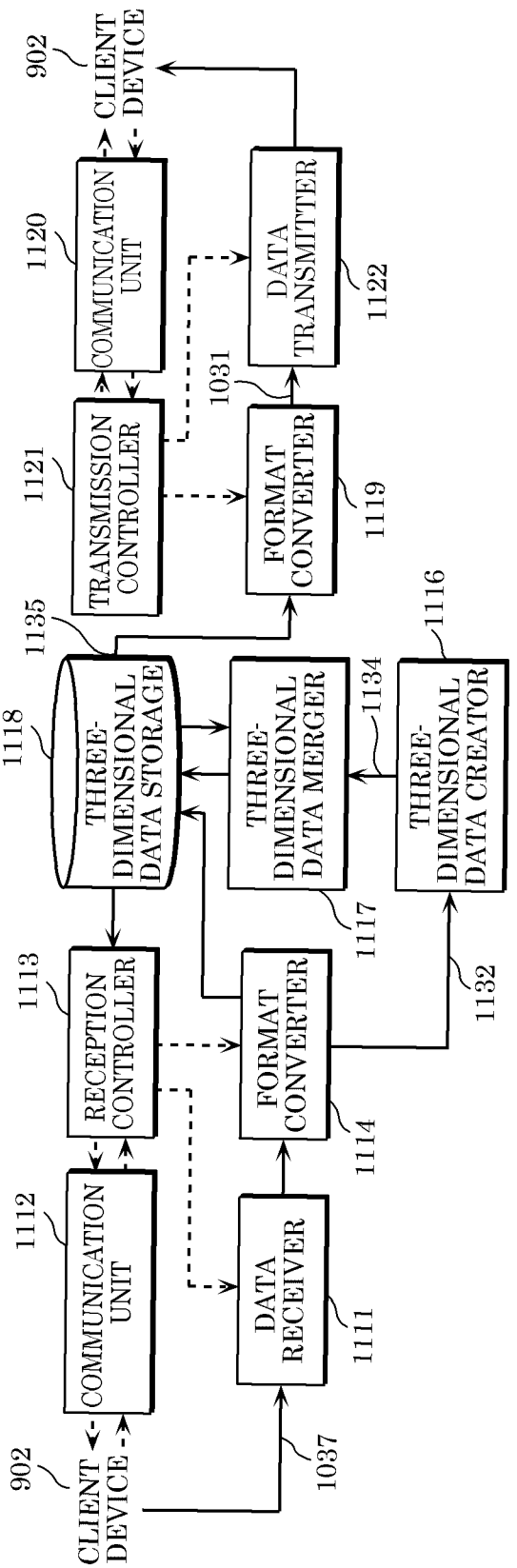

901
902
CLIENT DEVICE
1037
1111
DATA RECEIVER
1112
COMMUNICATION UNIT
1113
RECEPTION CONTROLLER
1114
FORMAT CONVERTER
1117
THREE-DIMENSIONAL DATA MERGER
1118
THREE-DIMENSIONAL DATA STORAGE
1121
TRANSMISSION CONTROLLER
1119
FORMAT CONVERTER
1122
DATA TRANSMITTER
1120
COMMUNICATION UNIT
902
CLIENT DEVICE
1132
1134
1135
1116
THREE-DIMENSIONAL DATA CREATOR
1031

FIG. 31

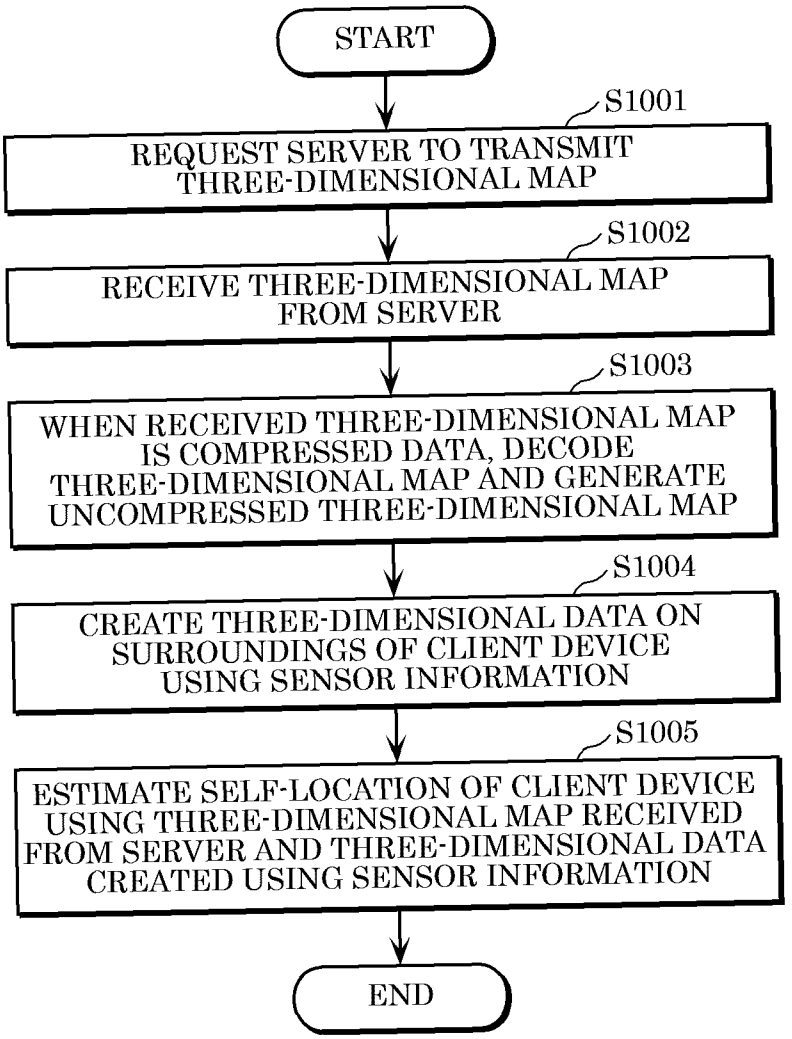

START
S1001
REQUEST SERVER TO TRANSMIT THREE-DIMENSIONAL MAP
S1002
RECEIVE THREE-DIMENSIONAL MAP FROM SERVER
S1003
WHEN RECEIVED THREE-DIMENSIONAL MAP IS COMPRESSED DATA, DECODE THREE-DIMENSIONAL MAP AND GENERATE UNCOMPRESSED THREE-DIMENSIONAL MAP
S1004
CREATE THREE-DIMENSIONAL DATA ON SURROUNDINGS OF CLIENT DEVICE USING SENSOR INFORMATION
S1005
ESTIMATE SELF-LOCATION OF CLIENT DEVICE USING THREE-DIMENSIONAL MAP RECEIVED FROM SERVER AND THREE-DIMENSIONAL DATA CREATED USING SENSOR INFORMATION
END

FIG. 32

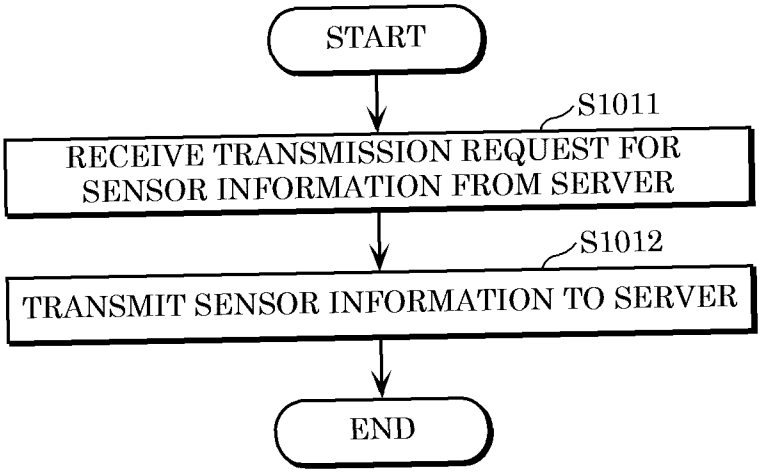

START
S1011
RECEIVE TRANSMISSION REQUEST FOR SENSOR INFORMATION FROM SERVER
S1012
TRANSMIT SENSOR INFORMATION TO SERVER
END

FIG. 33

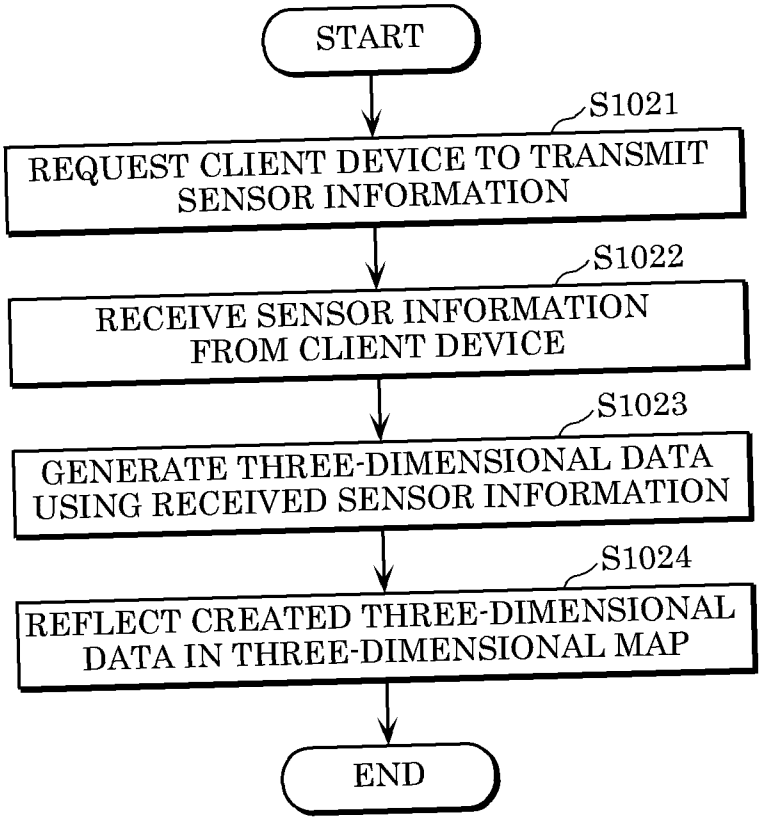

START
S1021
REQUEST CLIENT DEVICE TO TRANSMIT SENSOR INFORMATION
S1022
RECEIVE SENSOR INFORMATION FROM CLIENT DEVICE
S1023
GENERATE THREE-DIMENSIONAL DATA USING RECEIVED SENSOR INFORMATION
S1024
REFLECT CREATED THREE-DIMENSIONAL DATA IN THREE-DIMENSIONAL MAP
END

FIG. 34

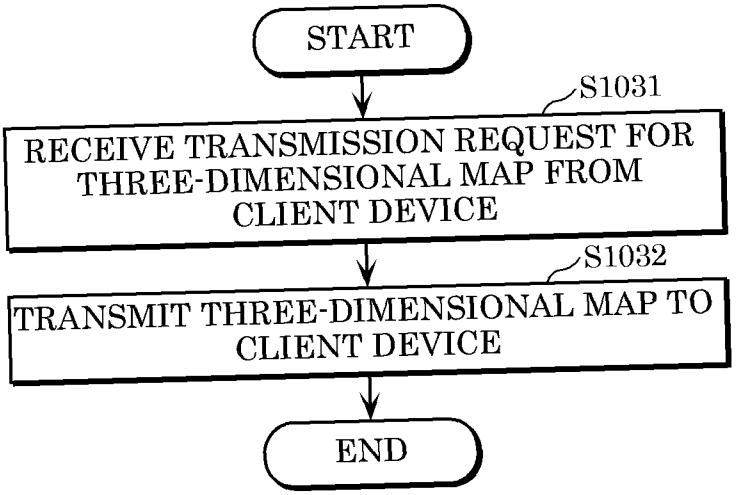

START
S1031
RECEIVE TRANSMISSION REQUEST FOR THREE-DIMENSIONAL MAP FROM CLIENT DEVICE
S1032
TRANSMIT THREE-DIMENSIONAL MAP TO CLIENT DEVICE
END

FIG. 35

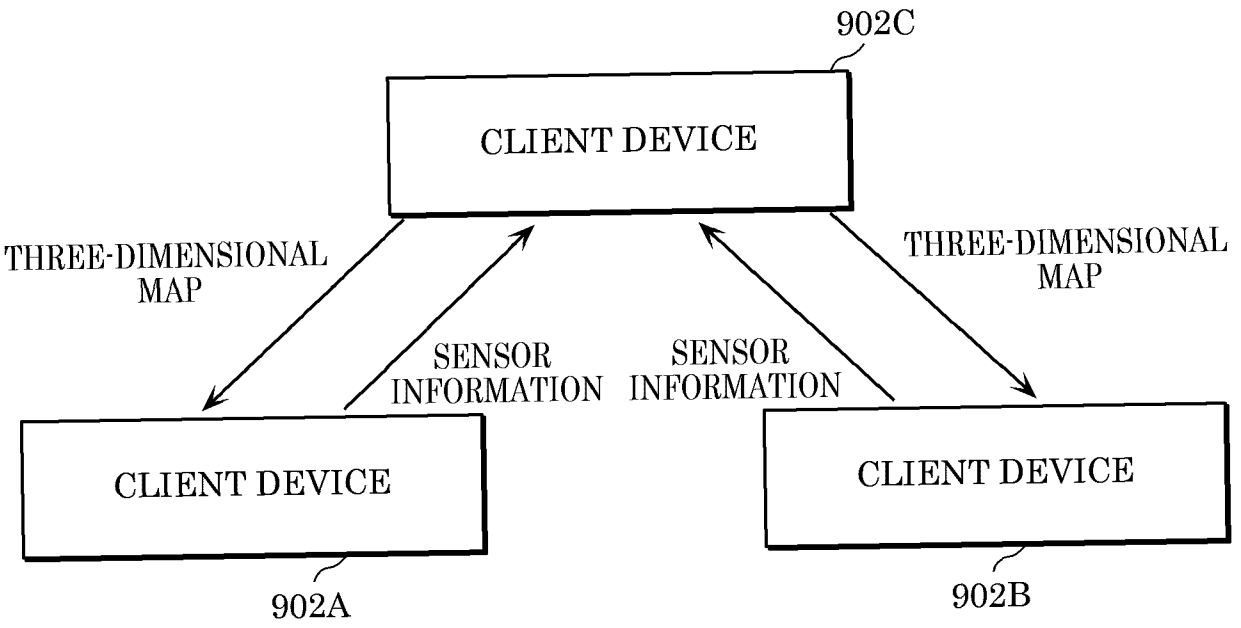

902C
CLIENT DEVICE
THREE-DIMENSIONAL MAP
THREE-DIMENSIONAL MAP
SENSOR INFORMATION
SENSOR INFORMATION
CLIENT DEVICE
CLIENT DEVICE
902A
902B

FIG. 36

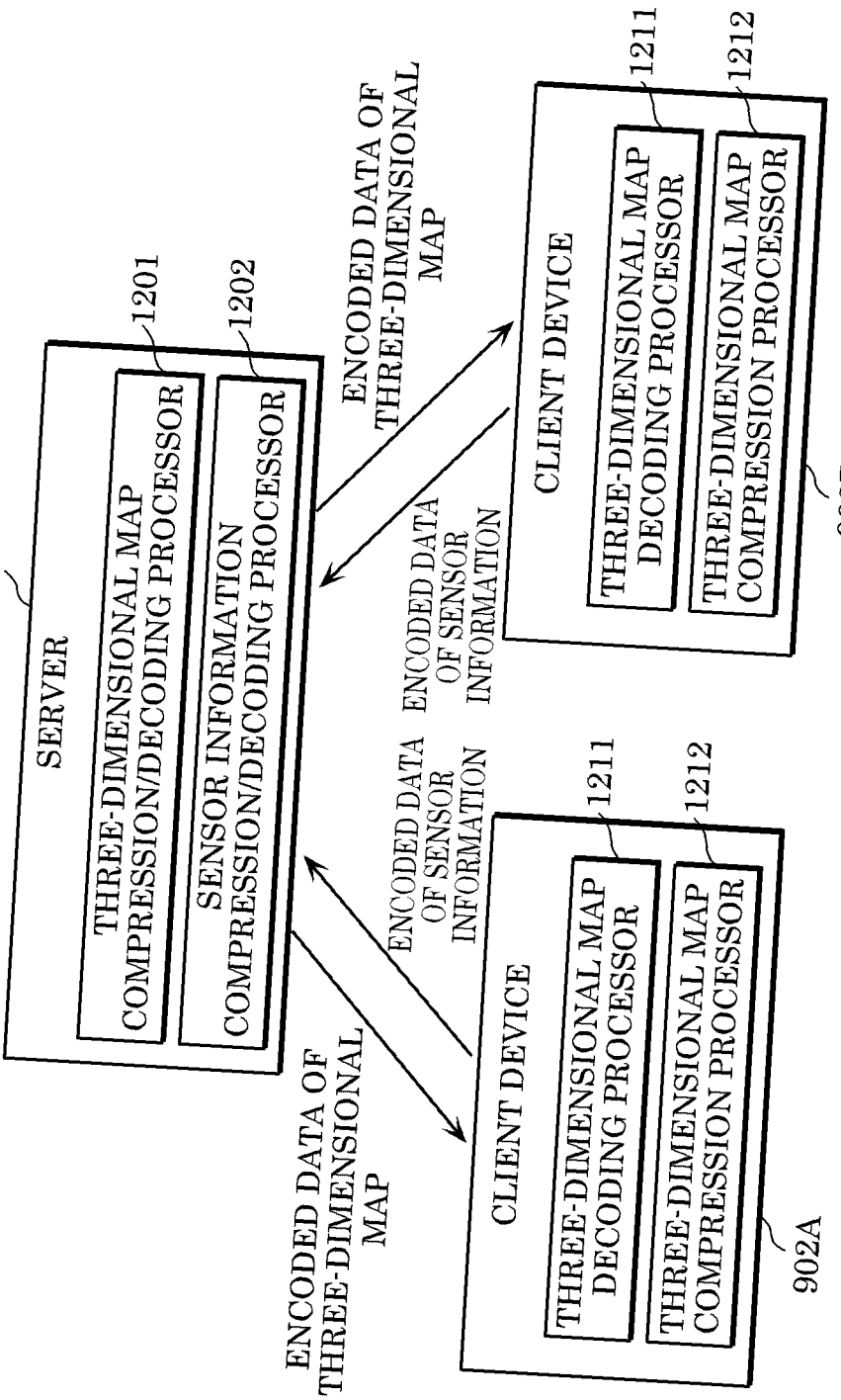

901
SERVER
THREE-DIMENSIONAL MAP COMPRESSION/DECODING PROCESSOR
1201
SENSOR INFORMATION COMPRESSION/DECODING PROCESSOR
1202
ENCODED DATA OF THREE-DIMENSIONAL MAP
ENCODED DATA OF SENSOR INFORMATION
ENCODED DATA OF SENSOR INFORMATION
ENCODED DATA OF THREE-DIMENSIONAL MAP
CLIENT DEVICE
THREE-DIMENSIONAL MAP DECODING PROCESSOR
1211
THREE-DIMENSIONAL MAP COMPRESSION PROCESSOR
1212
902A
CLIENT DEVICE
THREE-DIMENSIONAL MAP DECODING PROCESSOR
1211
THREE-DIMENSIONAL MAP COMPRESSION PROCESSOR
1212
902B

FIG. 37

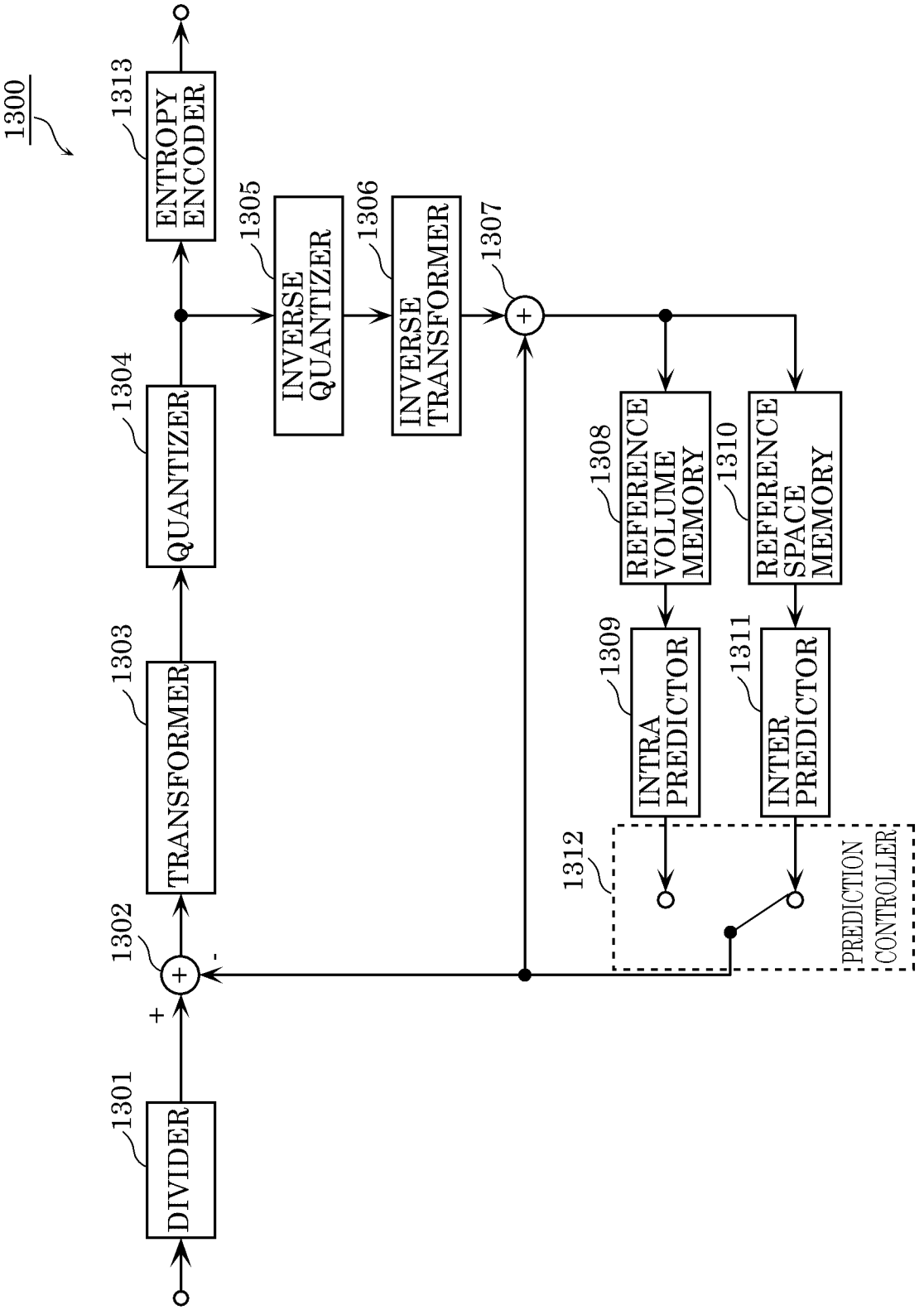

1300
1301
DIVIDER
1302
+
-
1303
TRANSFORMER
1304
QUANTIZER
1305
INVERSE QUANTIZER
1306
INVERSE TRANSFORMER
1307
1308
REFERENCE VOLUME MEMORY
1309
INTRA PREDICTOR
1310
REFERENCE SPACE MEMORY
1311
INTER PREDICTOR
1312
PREDICTION CONTROLLER
1313
ENTROPY ENCODER

FIG. 38

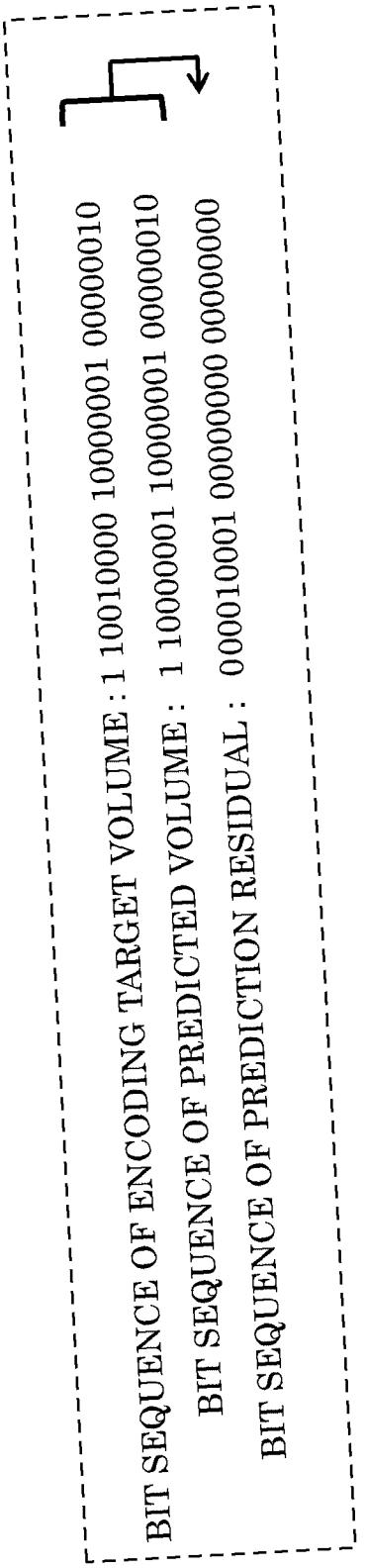

BIT SEQUENCE OF ENCODING TARGET VOLUME : 1 10010000 10000001 00000010
BIT SEQUENCE OF PREDICTED VOLUME : 1 10000001 10000001 00000010
BIT SEQUENCE OF PREDICTION RESIDUAL : 000010001 00000000 00000000

FIG. 39

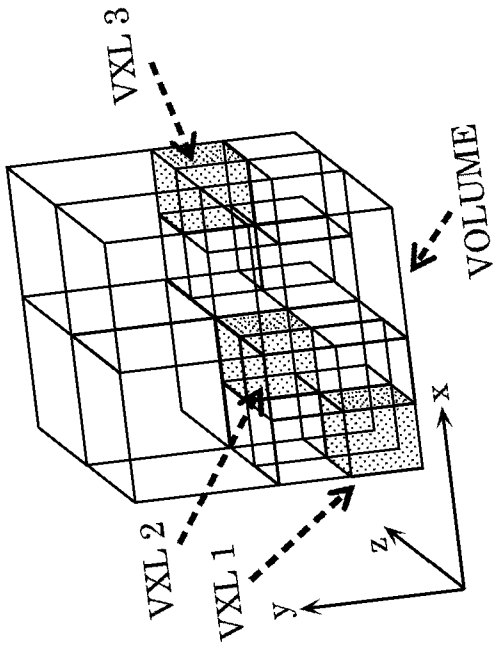


FIG. 40

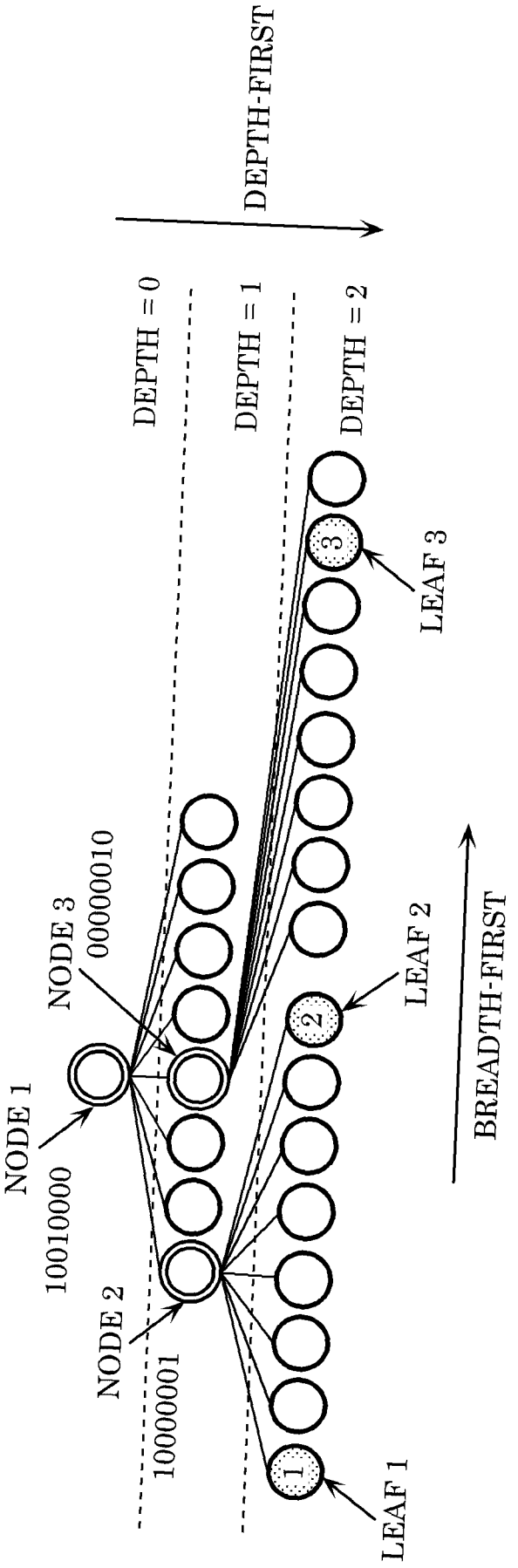
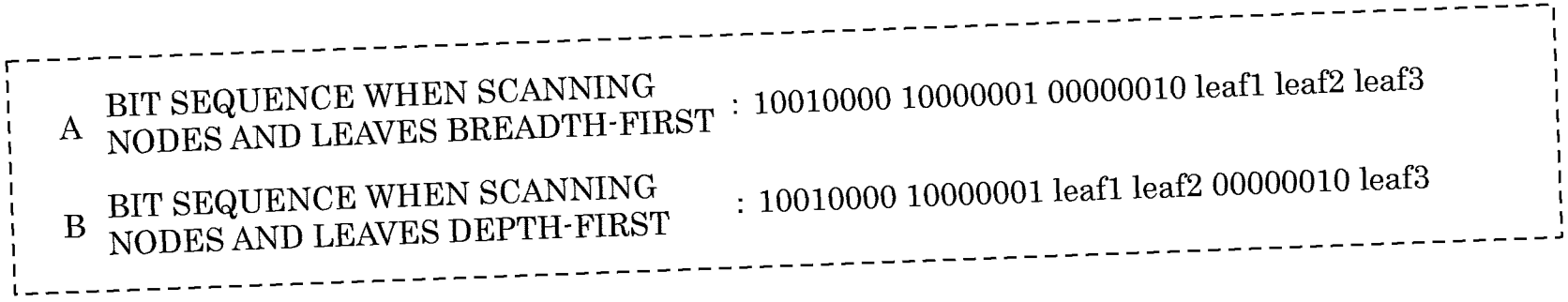

DEPTH-FIRST
DEPTH = 0
DEPTH = 1
DEPTH = 2
NODE 1
10010000
NODE 3
00000010
NODE 2
10000001
LEAF 1
LEAF 2
LEAF 3
BREADTH-FIRST

FIG. 41



CA 03090919 2020-08-10

FIG. 42

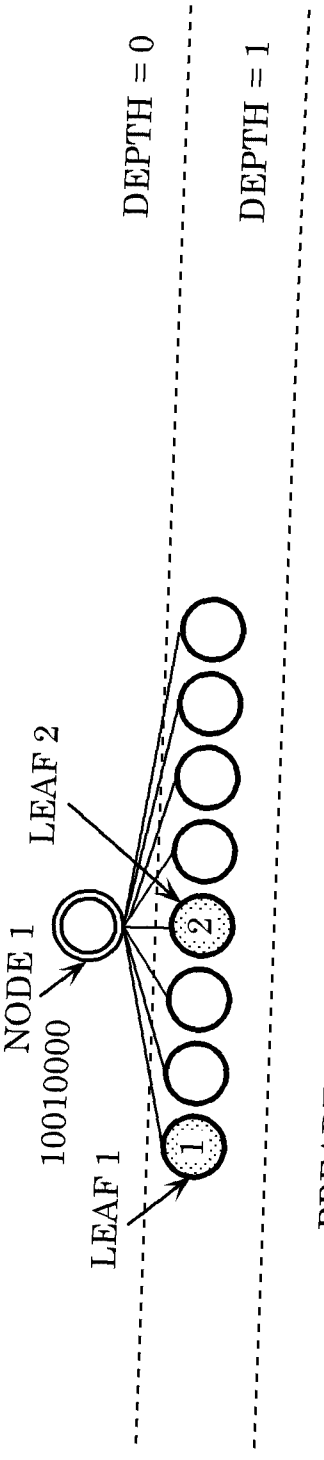

DEPTH = 0
DEPTH = 1
NODE 1
10010000
LEAF 2
LEAF 1
2
1
BREADTH-FIRST SCAN ORDER : 10010000

FIG. 43

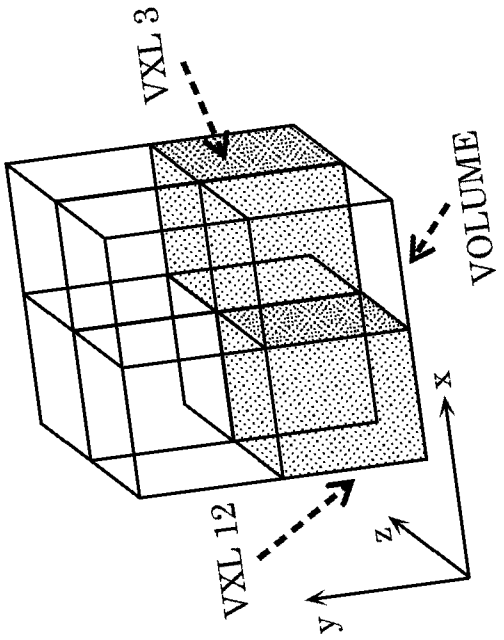


FIG. 44

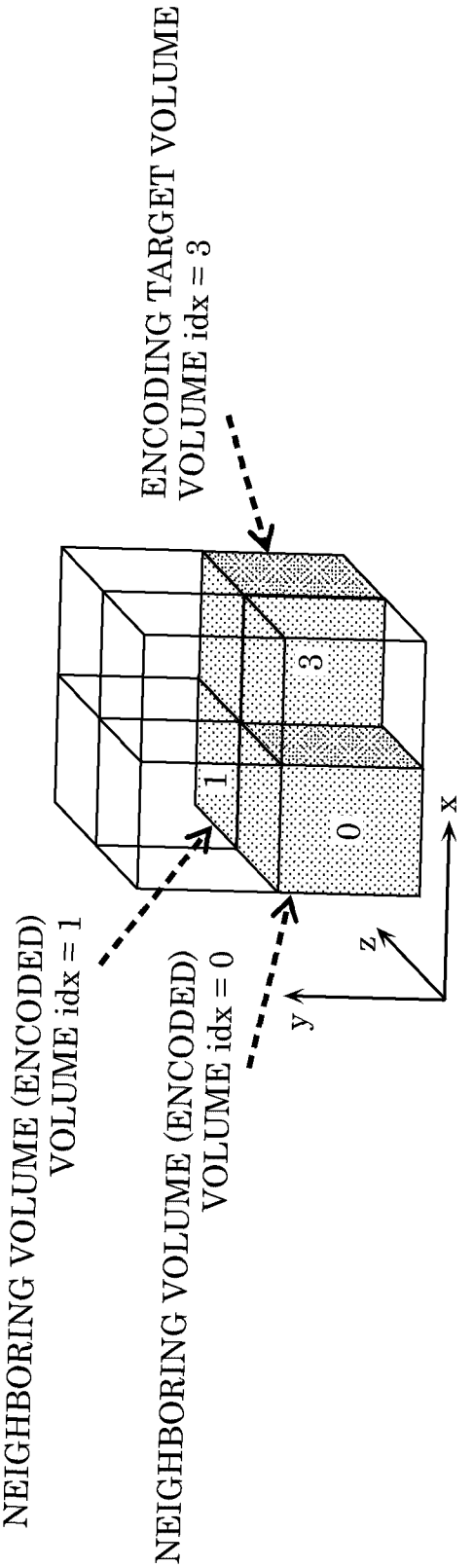

ENCODING TARGET VOLUME
VOLUME idx = 3
NEIGHBORING VOLUME (ENCODED)
VOLUME idx = 1
NEIGHBORING VOLUME (ENCODED)
VOLUME idx = 0
3
1
0
x
z
y

FIG. 45

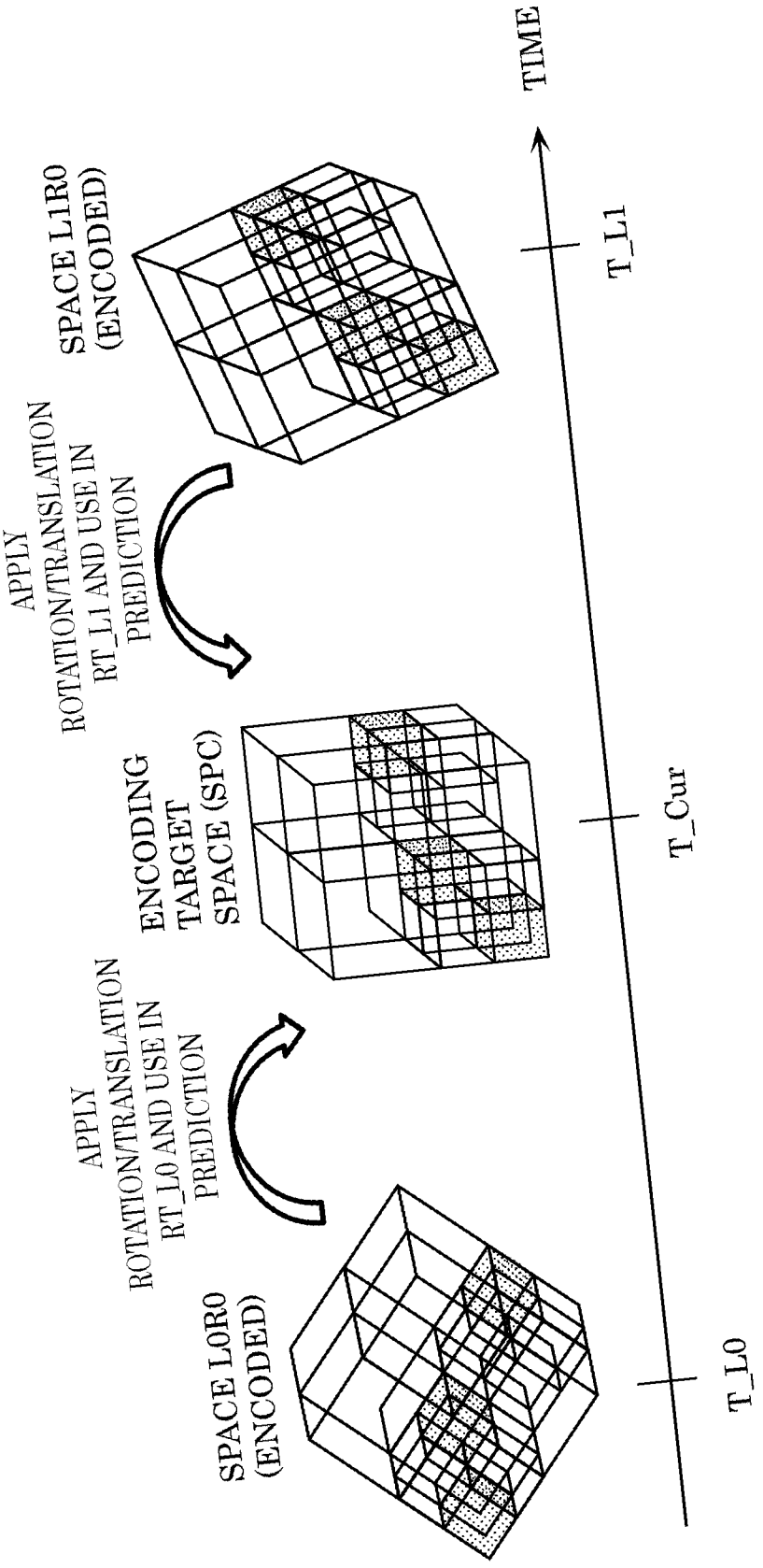

TIME
T_L0
T_Cur
T_L1
SPACE L0R0 (ENCODED)
APPLY ROTATION/TRANSLATION RT_L0 AND USE IN PREDICTION
ENCODING TARGET SPACE (SPC)
APPLY ROTATION/TRANSLATION RT_L1 AND USE IN PREDICTION
SPACE L1R0 (ENCODED)

FIG. 46

```
space_header() {
...
  for (i=0; i<MaxRefSpc_l0;i++) {
    RT_flag_l0[i]
    if (RT_flag_l0[i]) {
      R_l0[i]
      T_l0[i]
    }
  }
...
  for (i=0; i<MaxRefSpc_l1;i++) {
    RT_flag_l1[i]
    if (RT_flag_l1[i]) {
      R_l1[i]
      T_l1[i]
    }
  }
...
}
```


CA 03090919 2020-08-10

FIG. 47

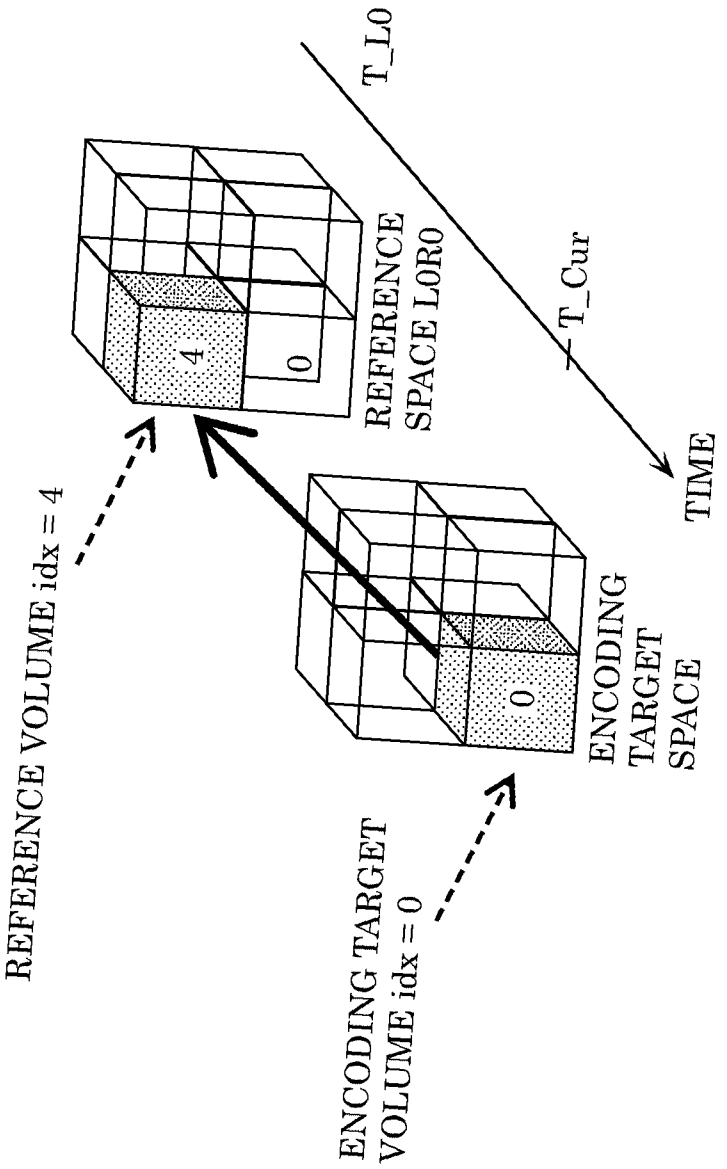

T_L0
T_Cur
TIME
REFERENCE SPACE L0R0
4
0
ENCODING TARGET SPACE
0
REFERENCE VOLUME idx = 4
ENCODING TARGET VOLUME idx = 0

FIG. 48

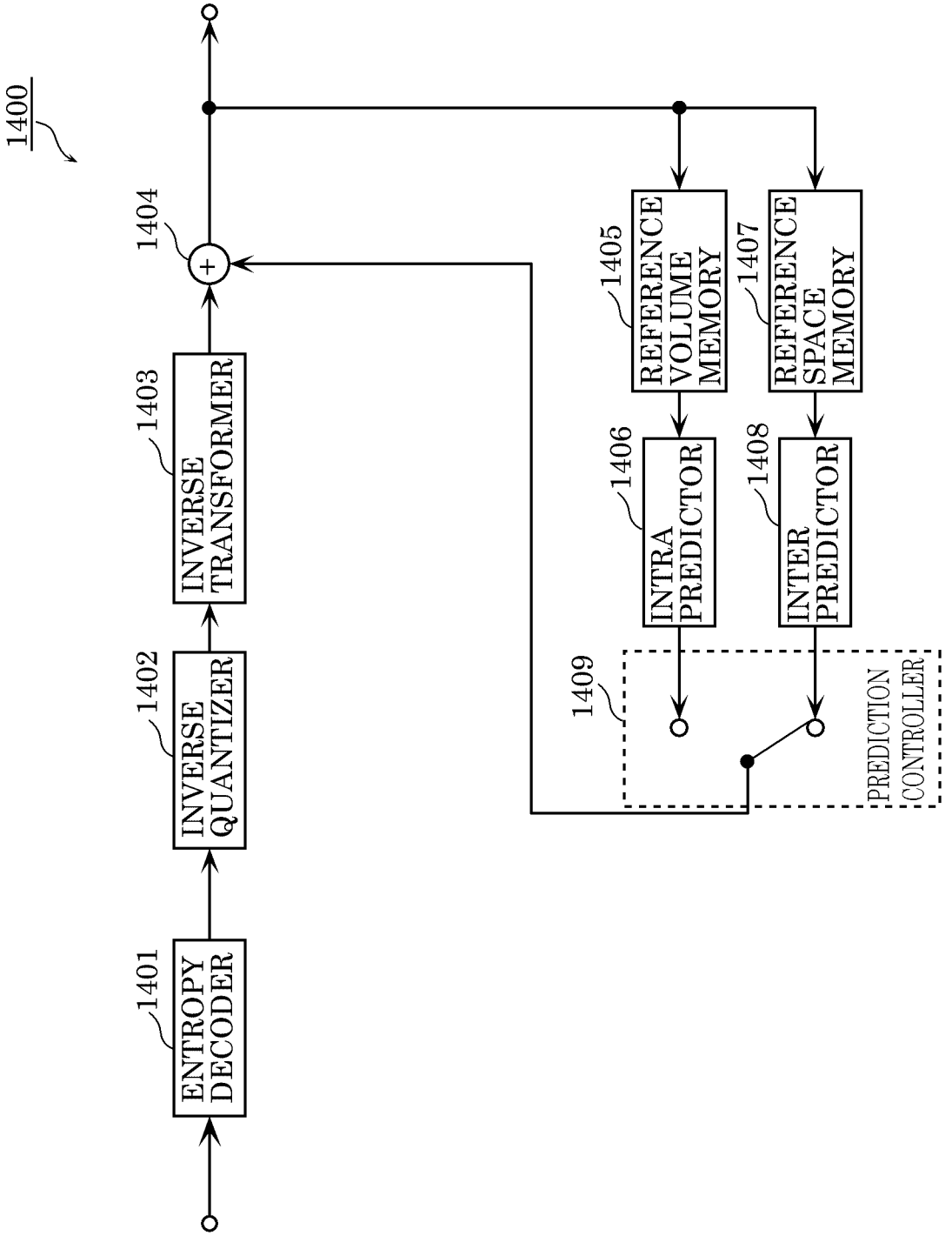

1400
1401
ENTROPY DECODER
1402
INVERSE QUANTIZER
1403
INVERSE TRANSFORMER
1404
+
1405
REFERENCE VOLUME MEMORY
1406
INTRA PREDICTOR
1407
REFERENCE SPACE MEMORY
1408
INTER PREDICTOR
1409
PREDICTION CONTROLLER

FIG. 49

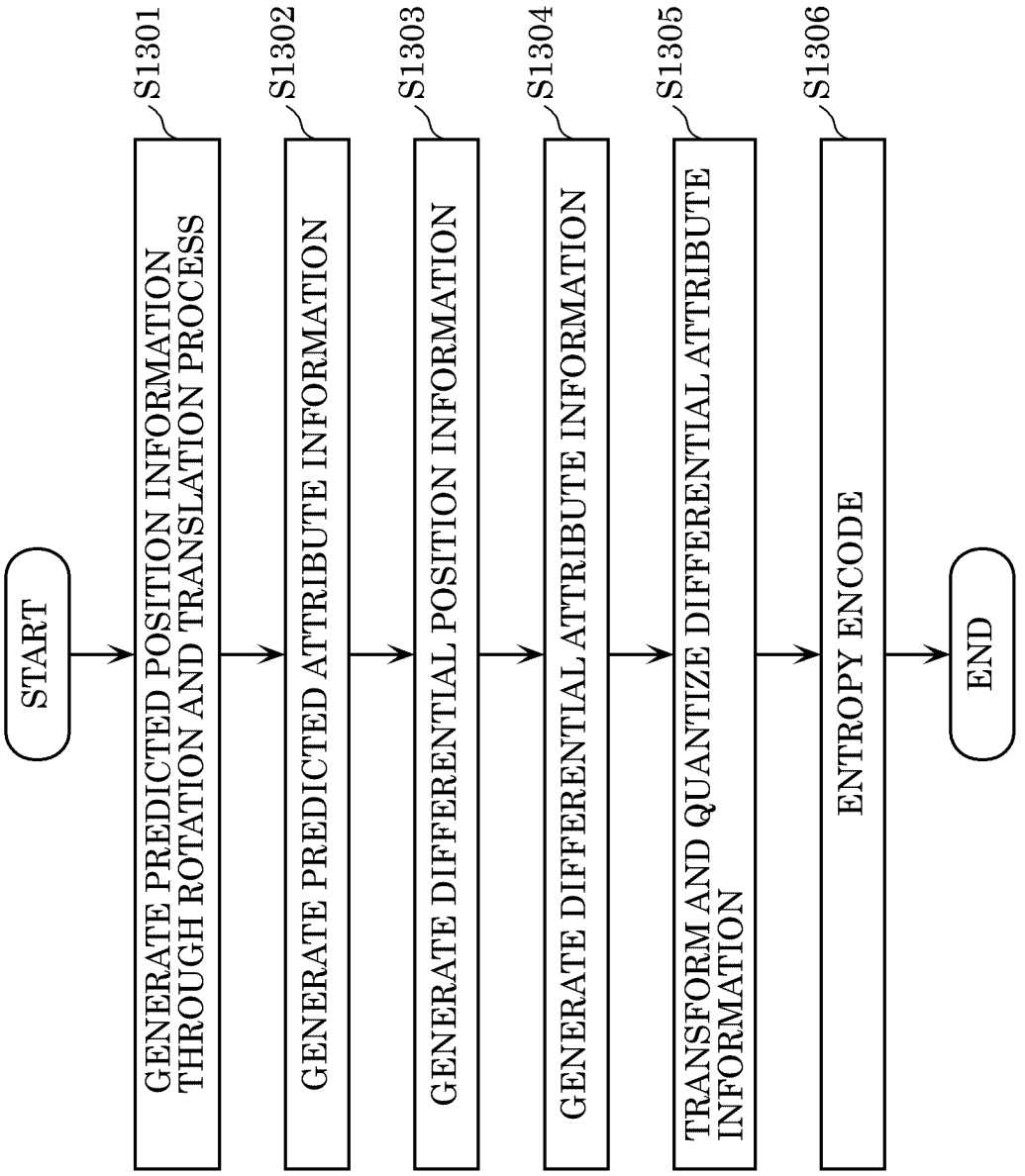

START
S1301
GENERATE PREDICTED POSITION INFORMATION THROUGH ROTATION AND TRANSLATION PROCESS
S1302
GENERATE PREDICTED ATTRIBUTE INFORMATION
S1303
GENERATE DIFFERENTIAL POSITION INFORMATION
S1304
GENERATE DIFFERENTIAL ATTRIBUTE INFORMATION
S1305
TRANSFORM AND QUANTIZE DIFFERENTIAL ATTRIBUTE INFORMATION
S1306
ENTROPY ENCODE
END

FIG. 50

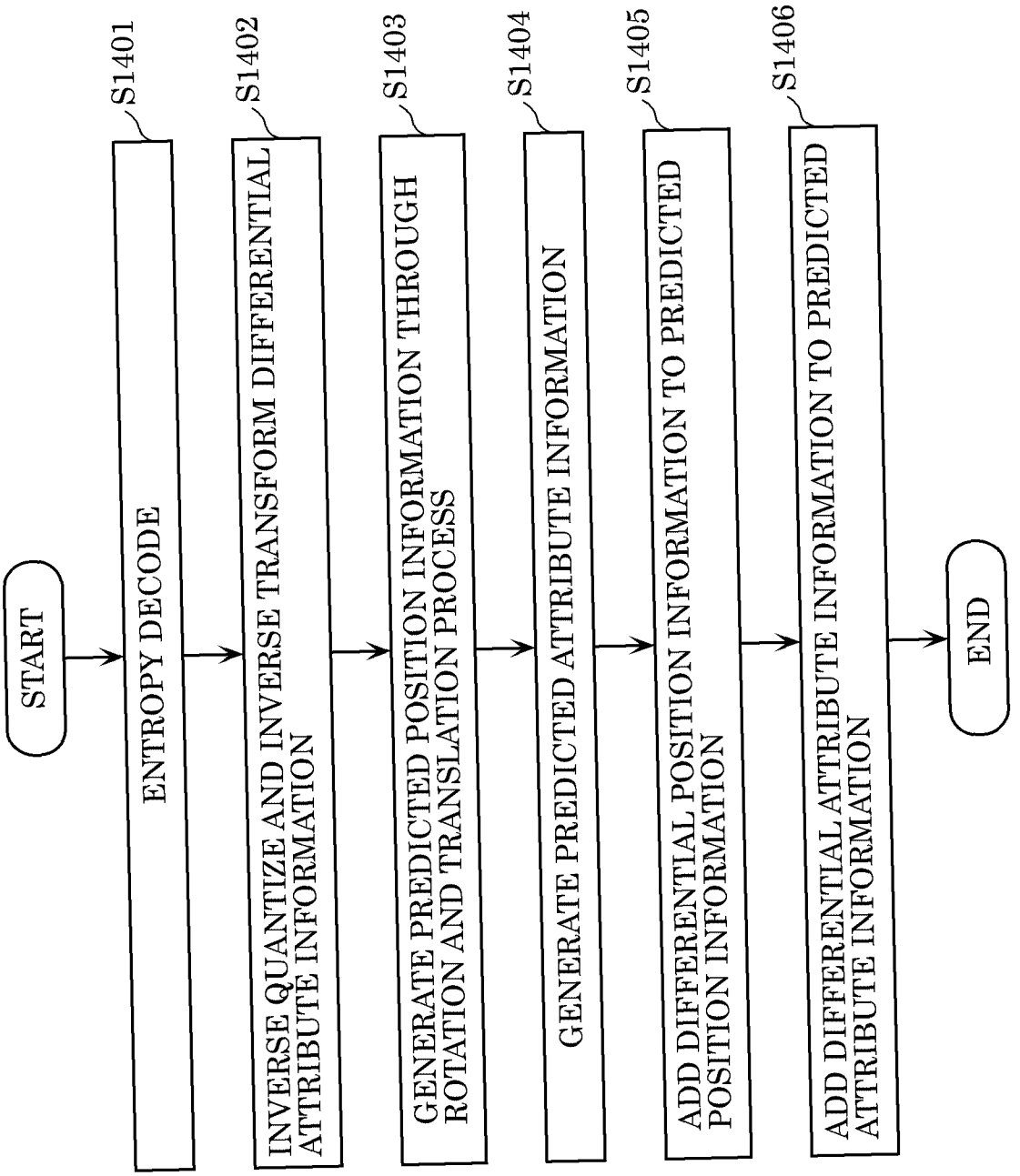

START
S1401
ENTROPY DECODE
S1402
INVERSE QUANTIZE AND INVERSE TRANSFORM DIFFERENTIAL ATTRIBUTE INFORMATION
S1403
GENERATE PREDICTED POSITION INFORMATION THROUGH ROTATION AND TRANSLATION PROCESS
S1404
GENERATE PREDICTED ATTRIBUTE INFORMATION
S1405
ADD DIFFERENTIAL POSITION INFORMATION TO PREDICTED POSITION INFORMATION
S1406
ADD DIFFERENTIAL ATTRIBUTE INFORMATION TO PREDICTED ATTRIBUTE INFORMATION
END

FIG. 51

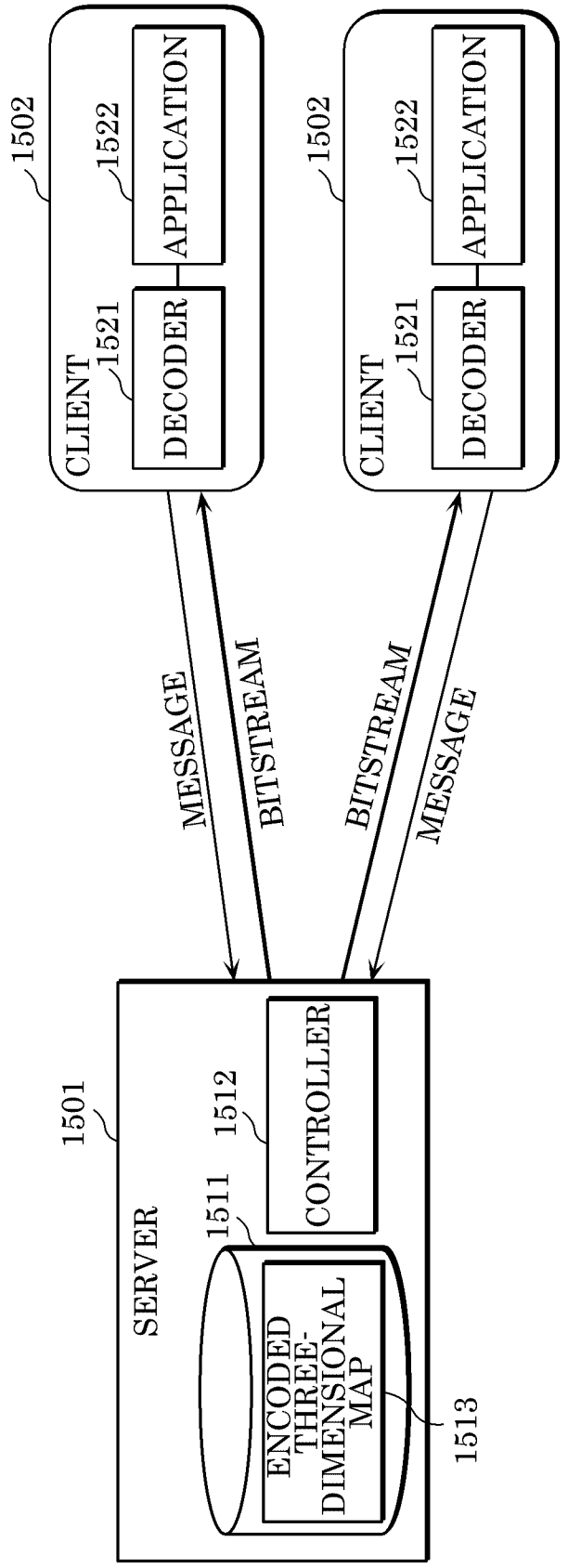

1502
CLIENT
1521
DECODER
1522
APPLICATION
1502
CLIENT
1521
DECODER
1522
APPLICATION
MESSAGE
BITSTREAM
BITSTREAM
MESSAGE
1501
SERVER
1512
CONTROLLER
1511
ENCODED THREE-DIMENSIONAL MAP
1513

FIG. 52

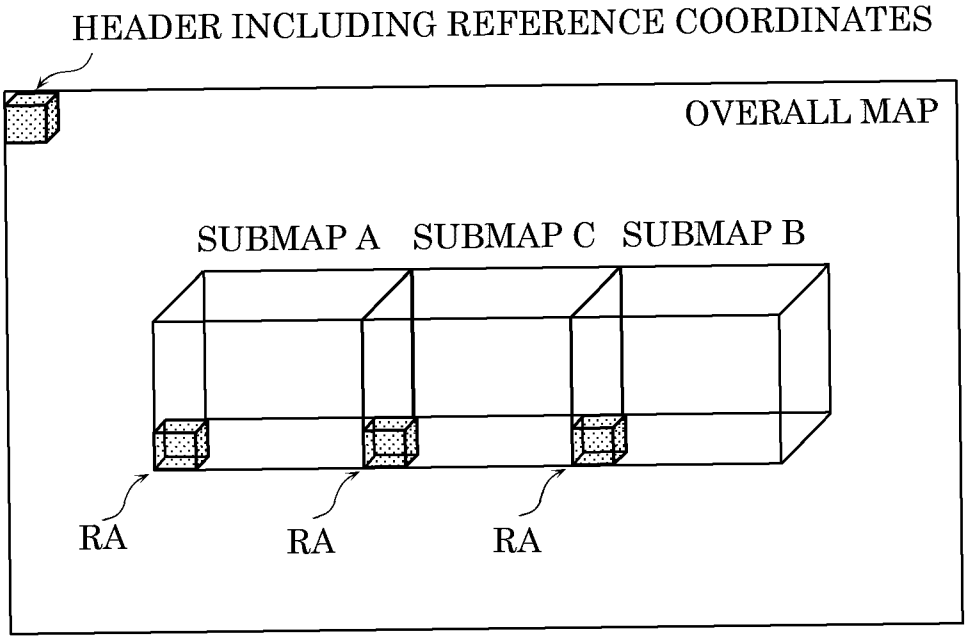

HEADER INCLUDING REFERENCE COORDINATES
OVERALL MAP
SUBMAP A
SUBMAP C
SUBMAP B
RA
RA
RA

FIG. 53

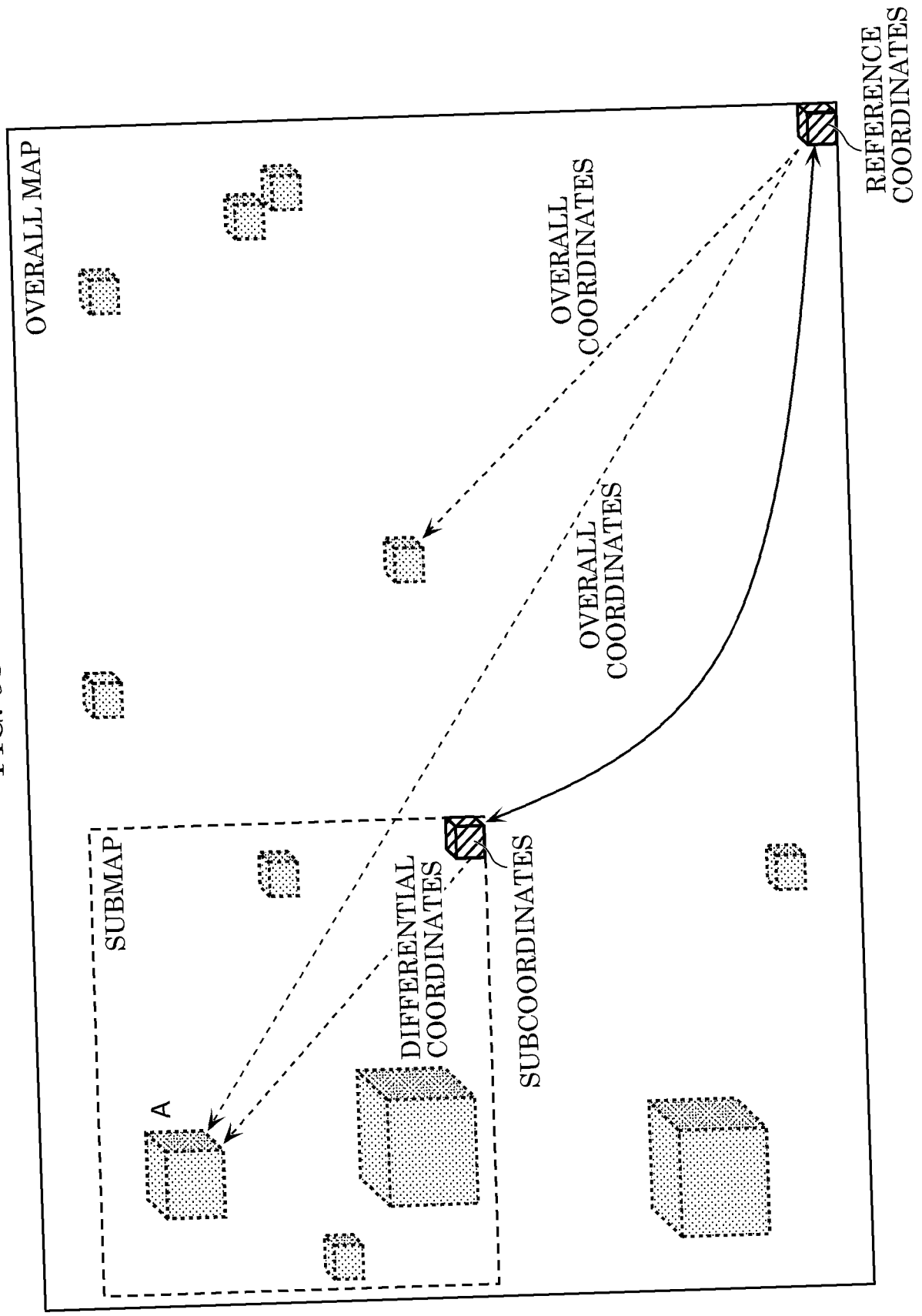

OVERALL MAP
REFERENCE COORDINATES
OVERALL COORDINATES
OVERALL COORDINATES
SUBMAP
DIFFERENTIAL COORDINATES
SUBCOORDINATES
A

FIG. 54

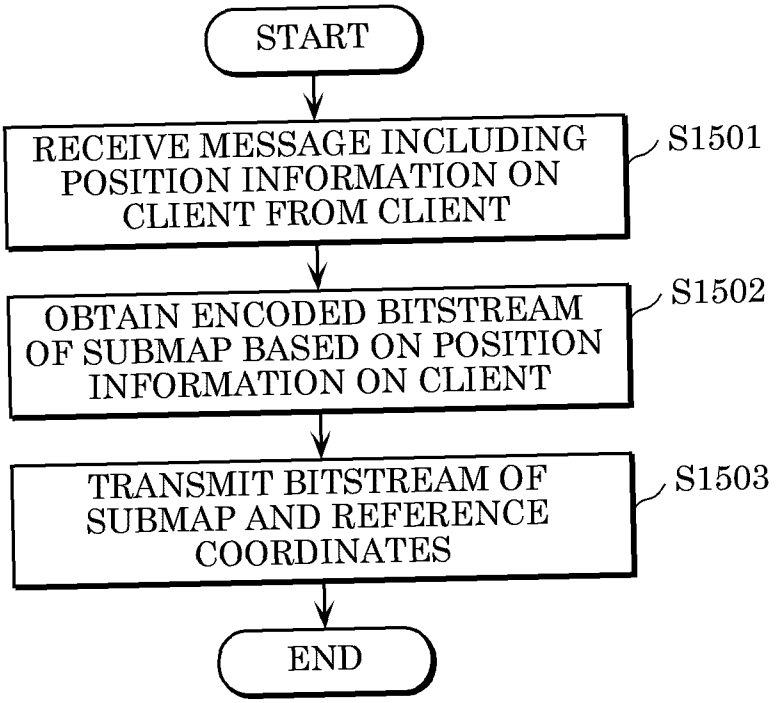

START
RECEIVE MESSAGE INCLUDING POSITION INFORMATION ON CLIENT FROM CLIENT
S1501
OBTAIN ENCODED BITSTREAM OF SUBMAP BASED ON POSITION INFORMATION ON CLIENT
S1502
TRANSMIT BITSTREAM OF SUBMAP AND REFERENCE COORDINATES
S1503
END

FIG. 55

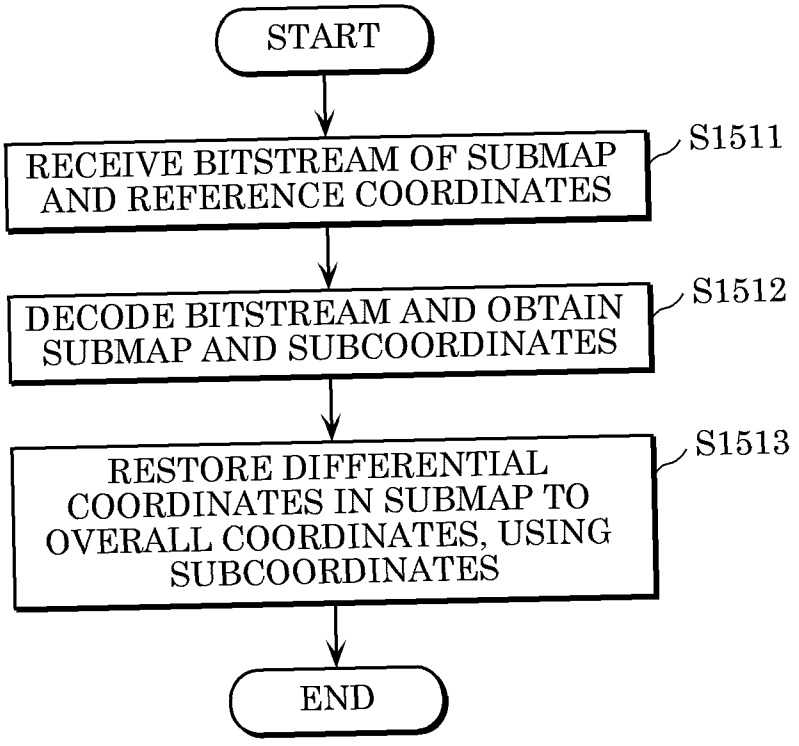

START
RECEIVE BITSTREAM OF SUBMAP AND REFERENCE COORDINATES
S1511
DECODE BITSTREAM AND OBTAIN SUBMAP AND SUBCOORDINATES
S1512
RESTORE DIFFERENTIAL COORDINATES IN SUBMAP TO OVERALL COORDINATES, USING SUBCOORDINATES
S1513
END

FIG. 56

```
sub-map_header() {
• • •
 NumOfPoint
 sub_coordinate_x
 sub_coordinate_y
 sub_coordinate_z
• • •
 sub_map_data()
• • •
}

 sub_map_data() {
• • •
 for (i=0; i<NumOfPoint; i++) {
  diff_x[i]
  diff_y[i]
  diff_z[i]
 }
• • •
 }
```

FIG. 57

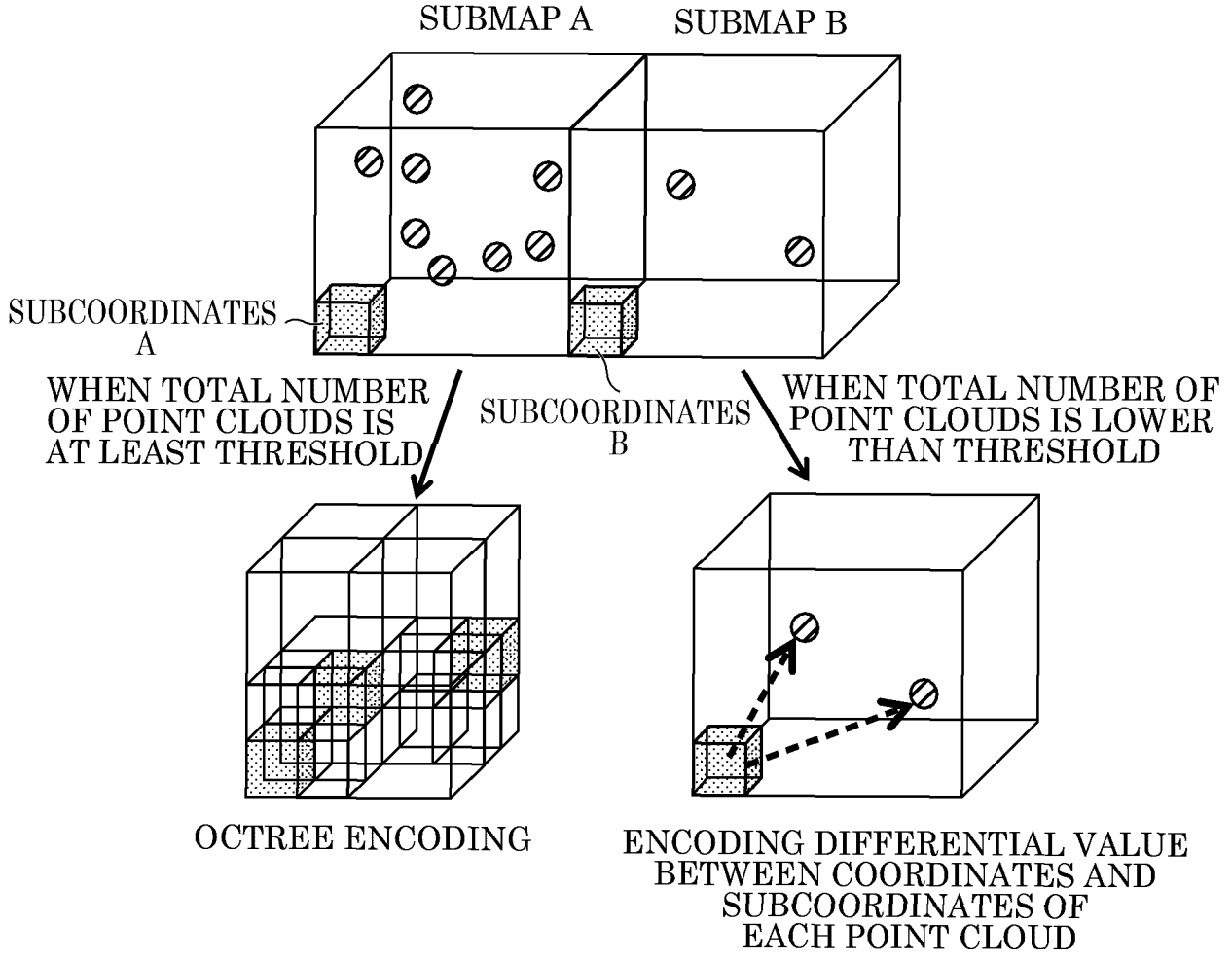

SUBMAP A
SUBMAP B
SUBCOORDINATES A
SUBCOORDINATES B
WHEN TOTAL NUMBER OF POINT CLOUDS IS AT LEAST THRESHOLD
WHEN TOTAL NUMBER OF POINT CLOUDS IS LOWER THAN THRESHOLD
OCTREE ENCODING
ENCODING DIFFERENTIAL VALUE BETWEEN COORDINATES AND SUBCOORDINATES OF EACH POINT CLOUD

FIG. 58

```
sub-map_header() {
 • • •
coding_type
If (coding_type == non_octree)
 {
  NumOfPoint
  sub_coordinate_x
  sub_coordinate_y
  sub_coordinate_z
 }
 else if (coding_type==octree) {
 {
  octree_info()
 }
 • • •
 sub_map_data()
 • • •
}

sub_map_data() {
 • • •
 if (coding_type==non_octree)
 {
  for (i=0; i<NumOfPoint; i++) {
   diff_x[i]
   diff_y[i]
   diff_z[i]
  }
 else (coding_type==octree) {
  octree_data()
 }
 • • •
}
```

FIG. 59

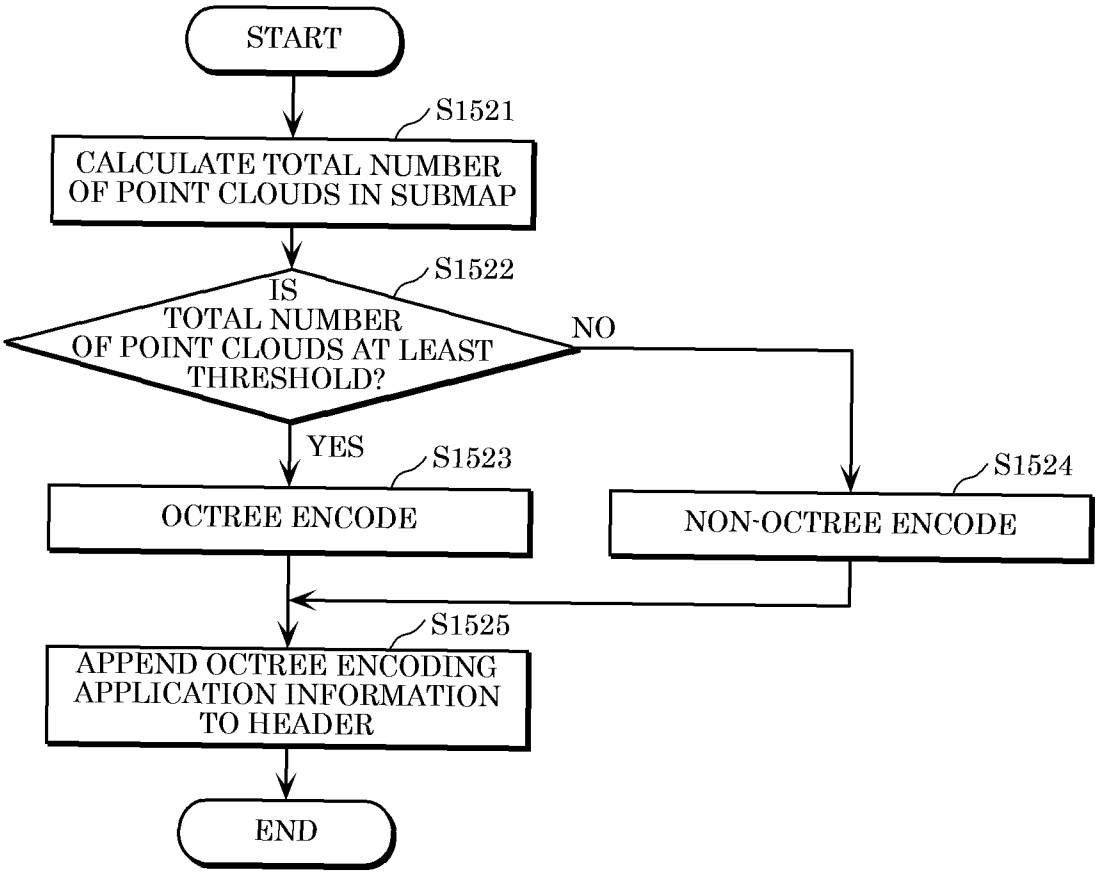

START
S1521
CALCULATE TOTAL NUMBER OF POINT CLOUDS IN SUBMAP
S1522
IS TOTAL NUMBER OF POINT CLOUDS AT LEAST THRESHOLD?
NO
YES
S1523
OCTREE ENCODE
S1524
NON-OCTREE ENCODE
S1525
APPEND OCTREE ENCODING APPLICATION INFORMATION TO HEADER
END

FIG. 60

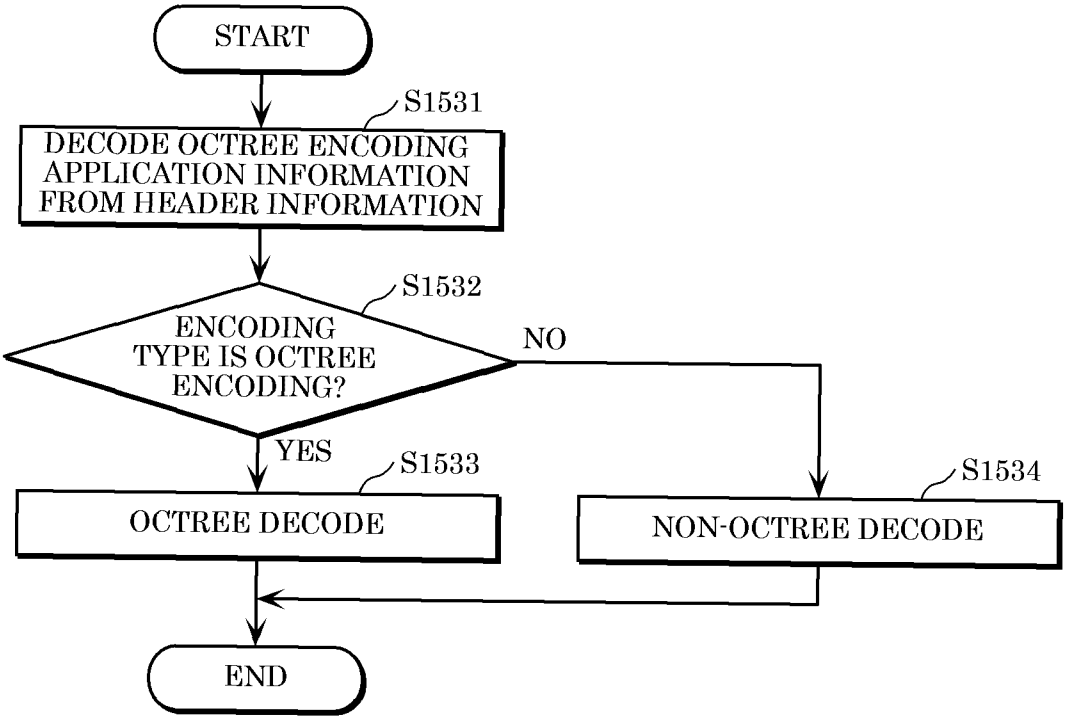

START
S1531
DECODE OCTREE ENCODING APPLICATION INFORMATION FROM HEADER INFORMATION
S1532
ENCODING TYPE IS OCTREE ENCODING?
NO
YES
S1533
OCTREE DECODE
S1534
NON-OCTREE DECODE
END

FIG. 61

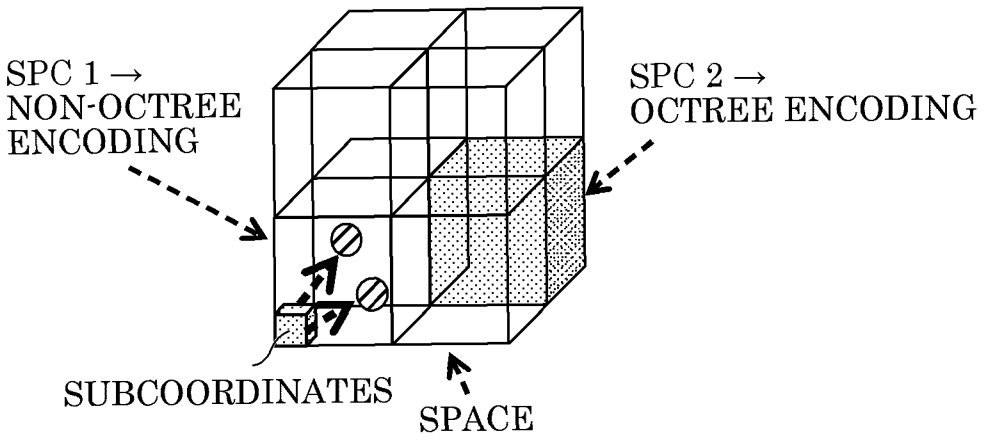


FIG. 62

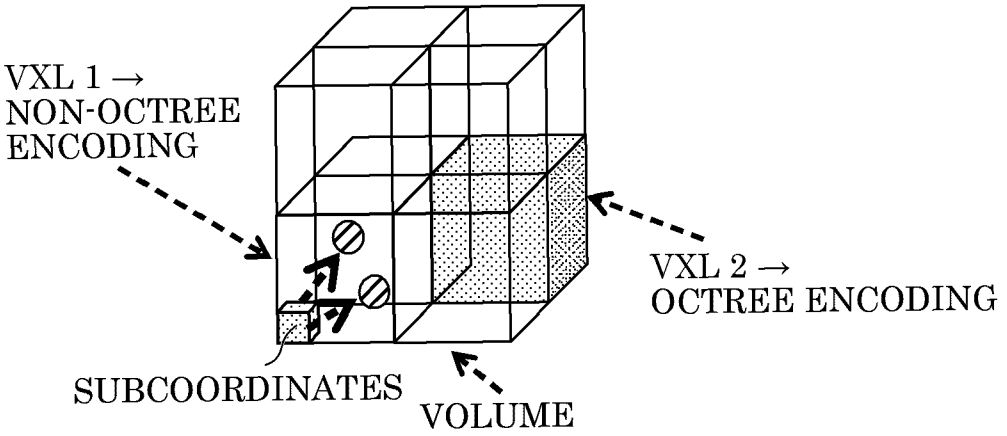


FIG. 63

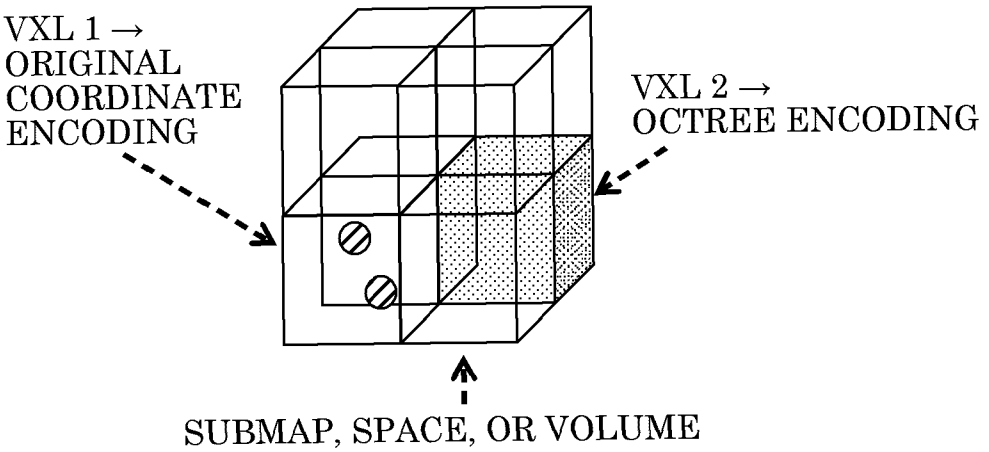


FIG. 64

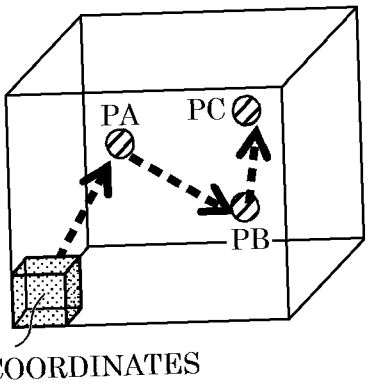


FIG. 65

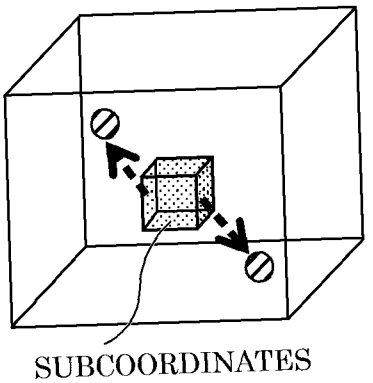


FIG. 66

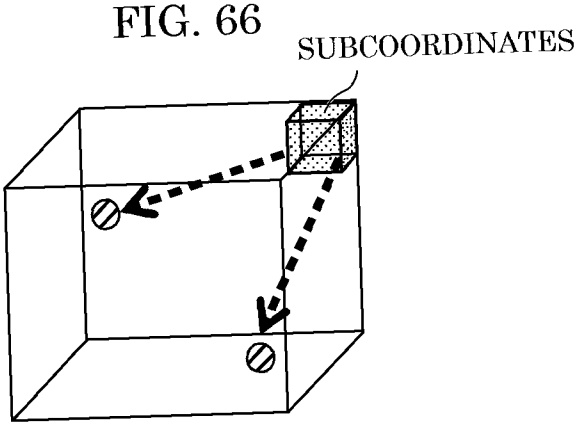


FIG. 67

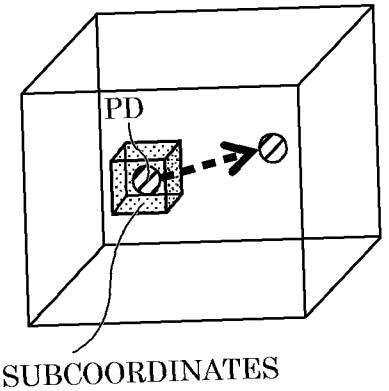


FIG. 68

```
volume_header() {
• • •
coding_type
If (coding_type == non_octree)
 {
  NumOfPoint
  sub_coordinate_x
  sub_coordinate_y
  sub_coordinate_z
 }
 else if (coding_type==octree) {
 {
   octree_info()
  }
• • •
 volume_data()
• • •
}

volume_data() {
• • •
 if (coding_type==non_octree)
 {
   for (i=0; i<NumOfPoint; i++) {
    diff_x[i]
    diff_y[i]
    diff_z[i]
   }
  else (coding_type==octree) {
   octree_data()
  }
• • •
}
```

FIG. 69

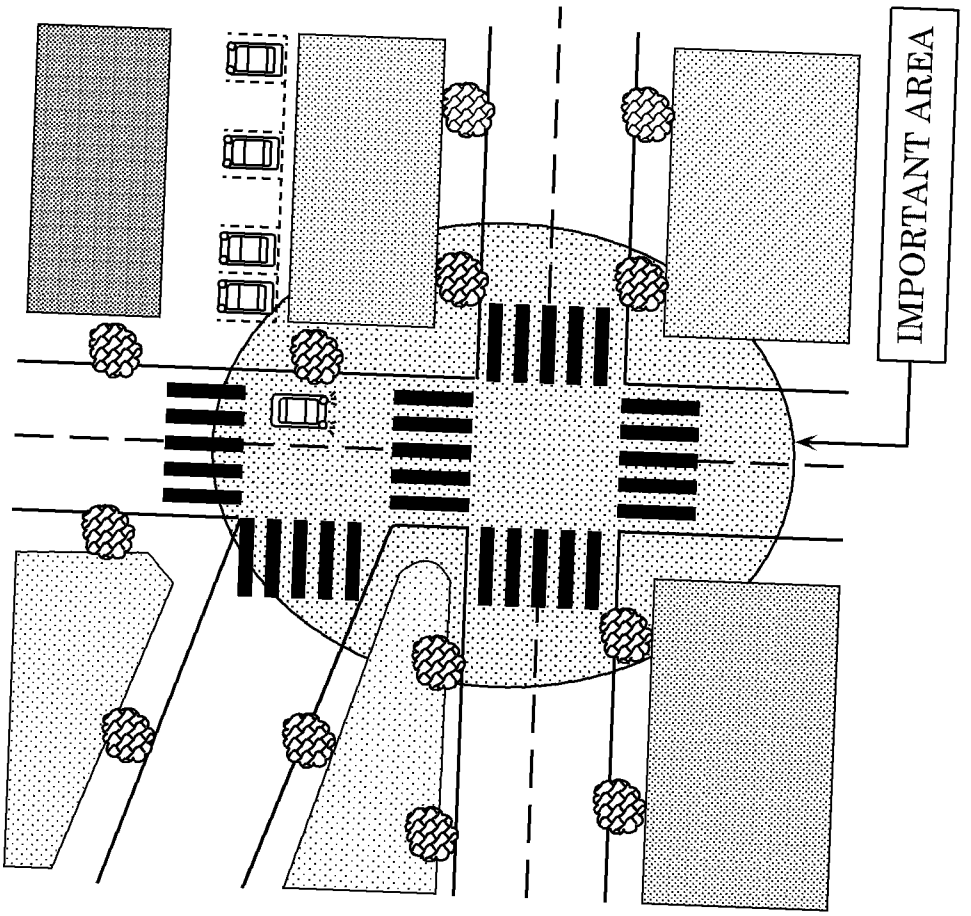


FIG. 70

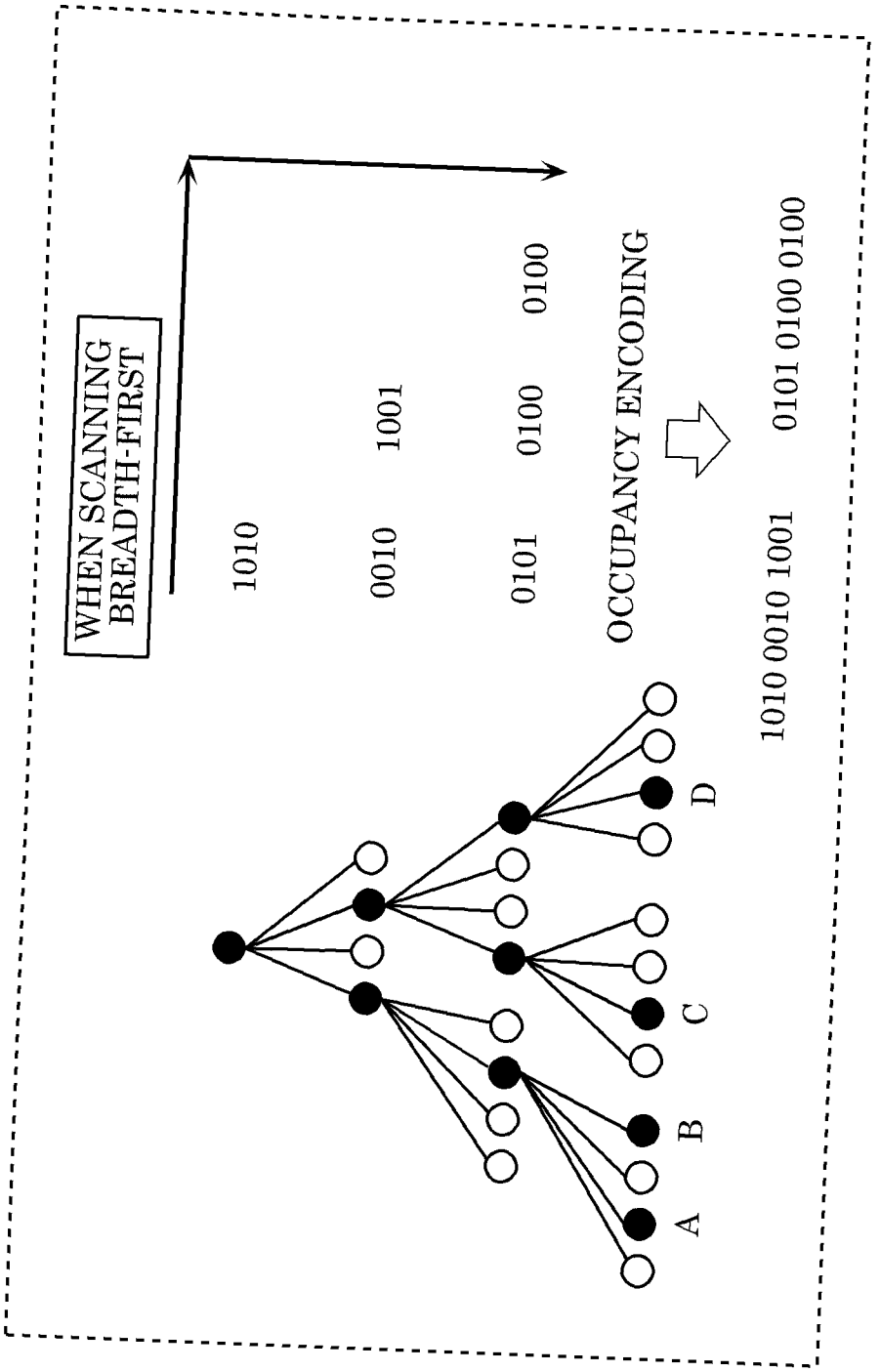

WHEN SCANNING
BREADTH-FIRST
1010
0010
1001
0101
0100
0100
OCCUPANCY ENCODING
1010 0010 1001
0101 0100 0100
A
B
C
D

FIG. 71

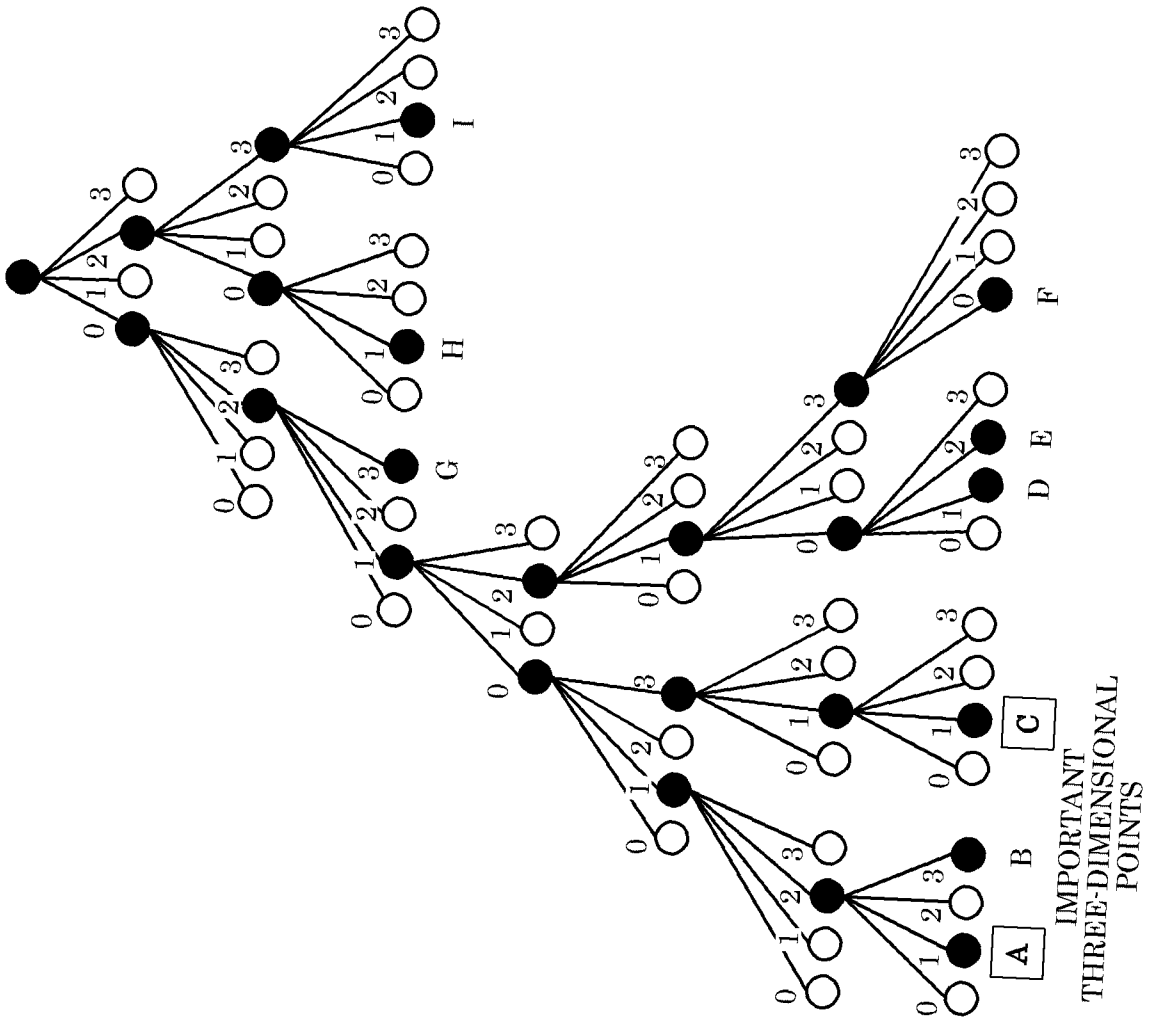


FIG. 72

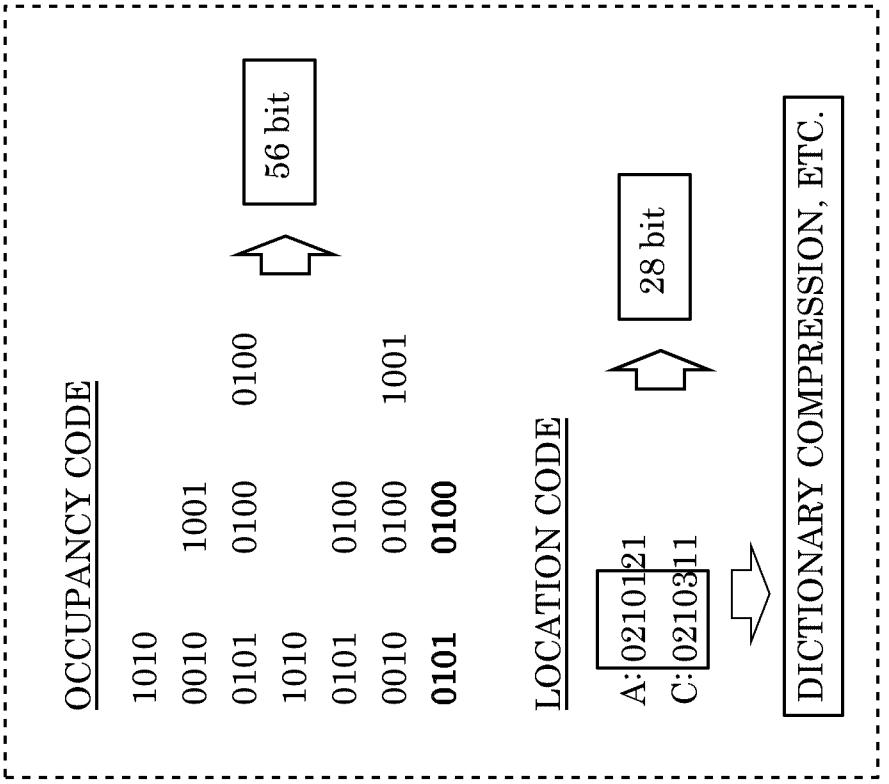

OCCUPANCY CODE
1010
0010 1001
0101 0100 0100
1010
0101 0100
0010 0100 1001
0101 0100
56 bit
LOCATION CODE
A: 02101 21
C: 02103 11
28 bit
DICTIONARY COMPRESSION, ETC.

FIG. 73

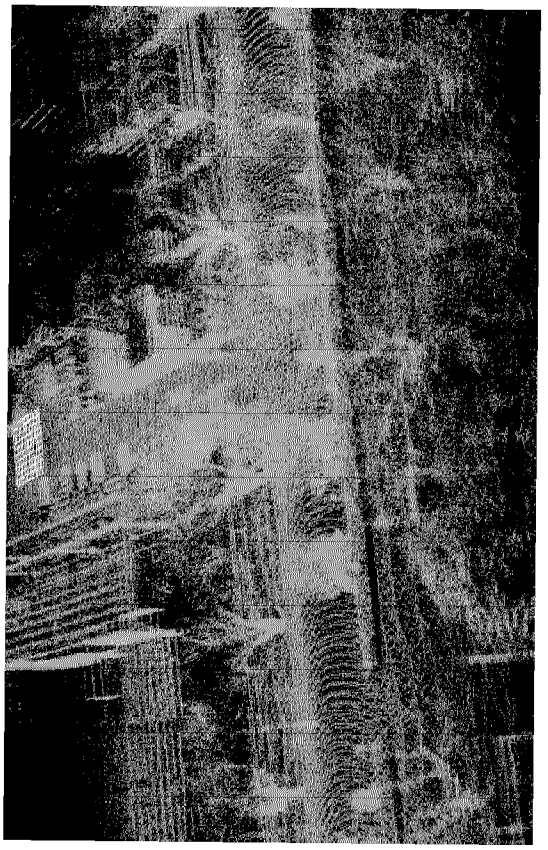


FIG. 74

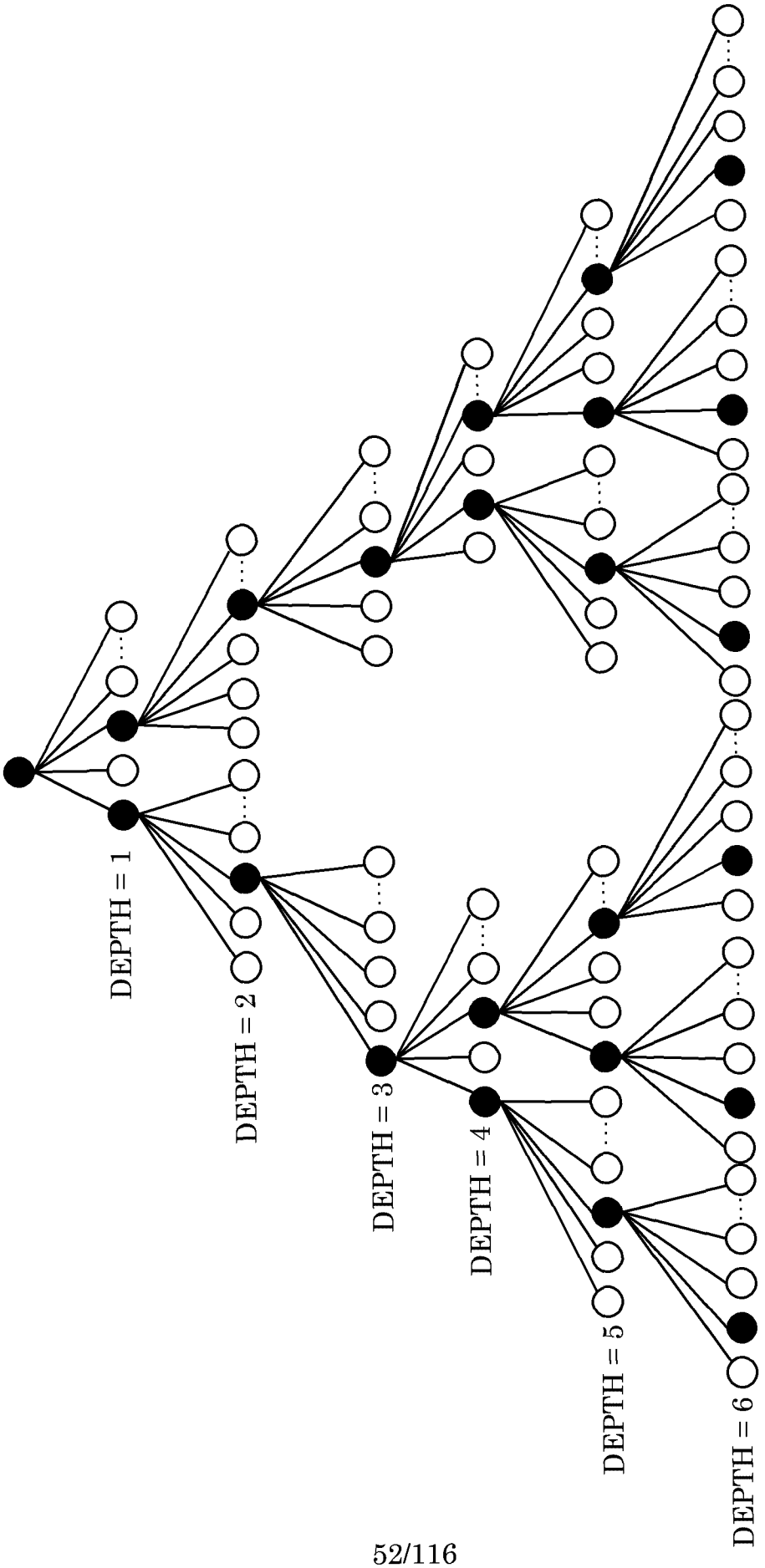


FIG. 75

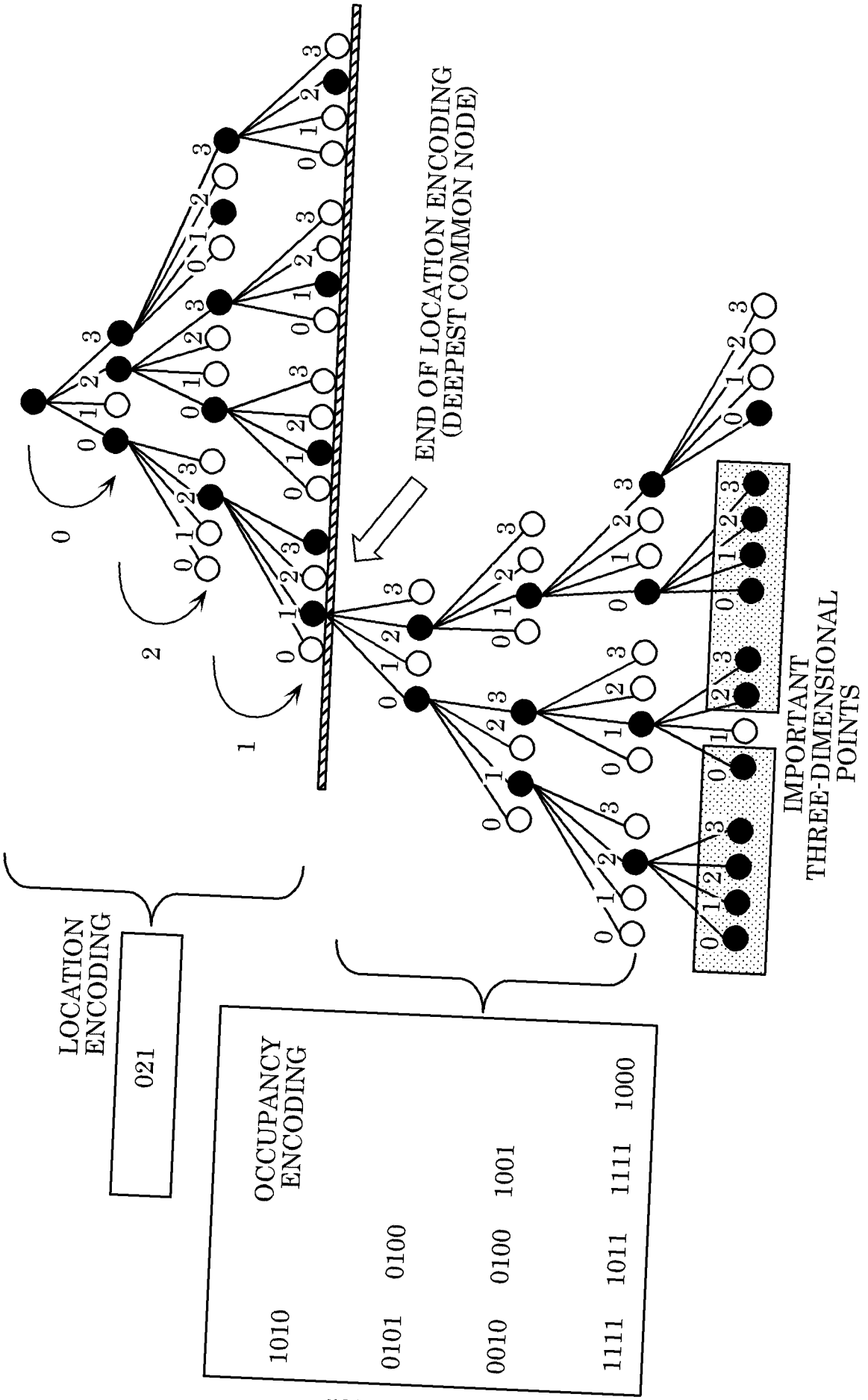

LOCATION ENCODING
021
OCCUPANCY ENCODING
1010
0101 0100
0010 0100 1001
1111 1011 1111 1000
END OF LOCATION ENCODING (DEEPEST COMMON NODE)
IMPORTANT THREE-DIMENSIONAL POINTS
0
1
2

FIG. 76

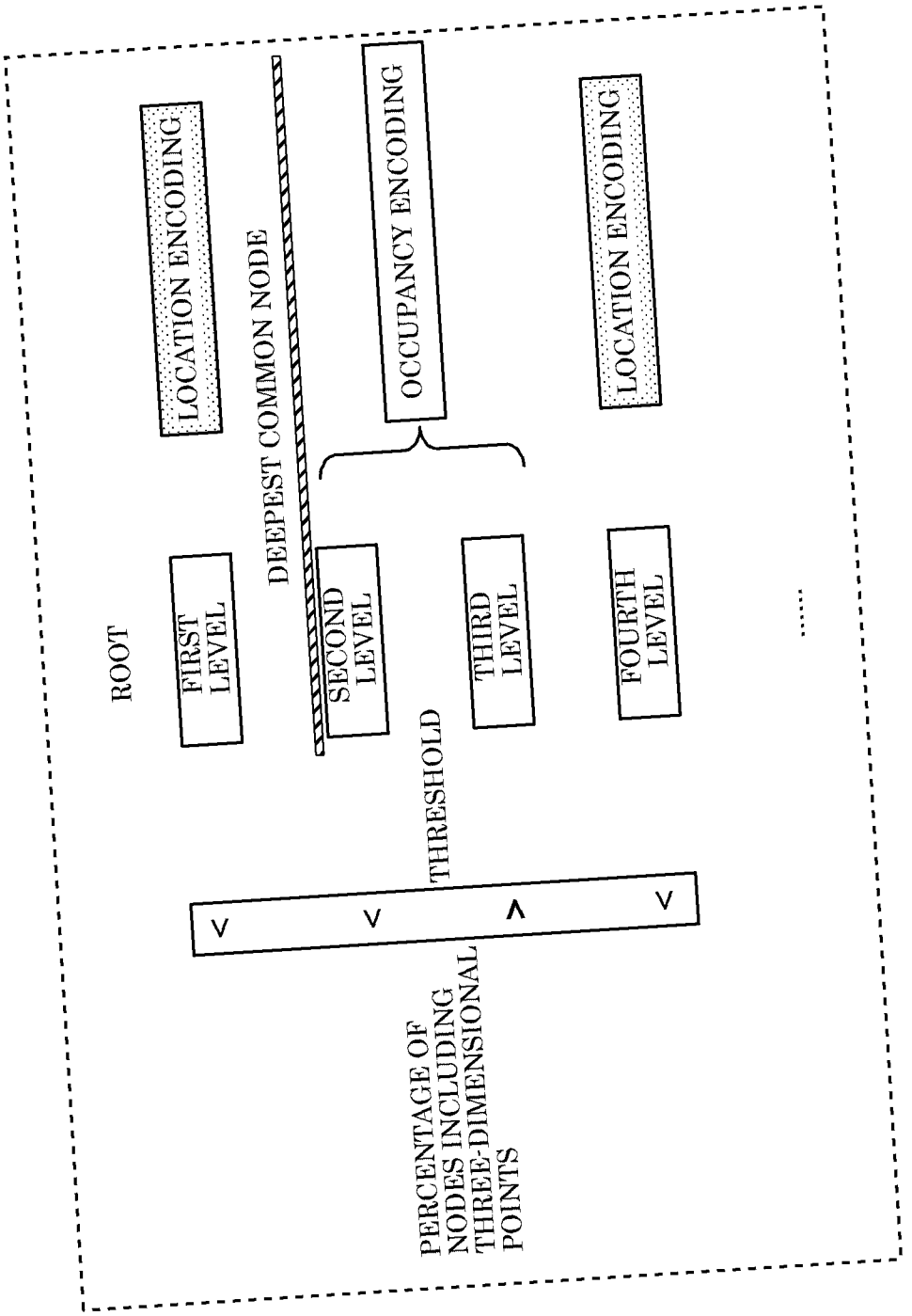

ROOT
FIRST LEVEL
SECOND LEVEL
THIRD LEVEL
FOURTH LEVEL
.....
LOCATION ENCODING
DEEPEST COMMON NODE
OCCUPANCY ENCODING
LOCATION ENCODING
THRESHOLD
PERCENTAGE OF NODES INCLUDING THREE-DIMENSIONAL POINTS

FIG. 77

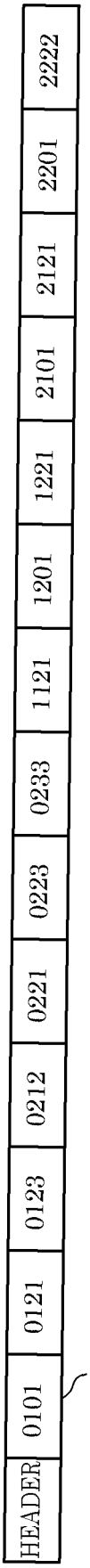

HEADER
0101
0121
0123
0212
0221
0223
0233
1121
1201
1221
2101
2121
2201
2222
LOCATION CODE

FIG. 78

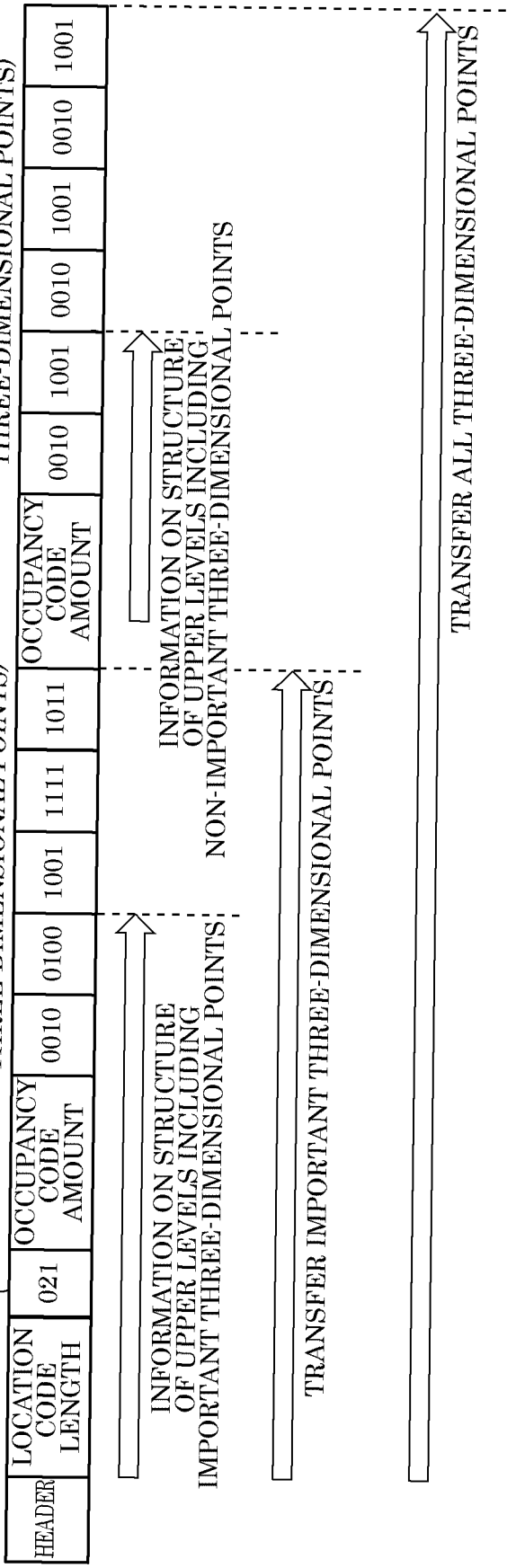

HEADER
LOCATION CODE LENGTH
021
OCCUPANCY CODE AMOUNT
LOCATION CODE
OCCUPANCY CODE (IMPORTANT THREE-DIMENSIONAL POINTS)
0010
0100
1001
1111
1011
OCCUPANCY CODE AMOUNT
OCCUPANCY CODE (NON-IMPORTANT THREE-DIMENSIONAL POINTS)
0010
1001
0010
1001
0010
1001
INFORMATION ON STRUCTURE OF UPPER LEVELS INCLUDING IMPORTANT THREE-DIMENSIONAL POINTS
INFORMATION ON STRUCTURE OF UPPER LEVELS INCLUDING NON-IMPORTANT THREE-DIMENSIONAL POINTS
TRANSFER IMPORTANT THREE-DIMENSIONAL POINTS
TRANSFER ALL THREE-DIMENSIONAL POINTS

FIG. 79

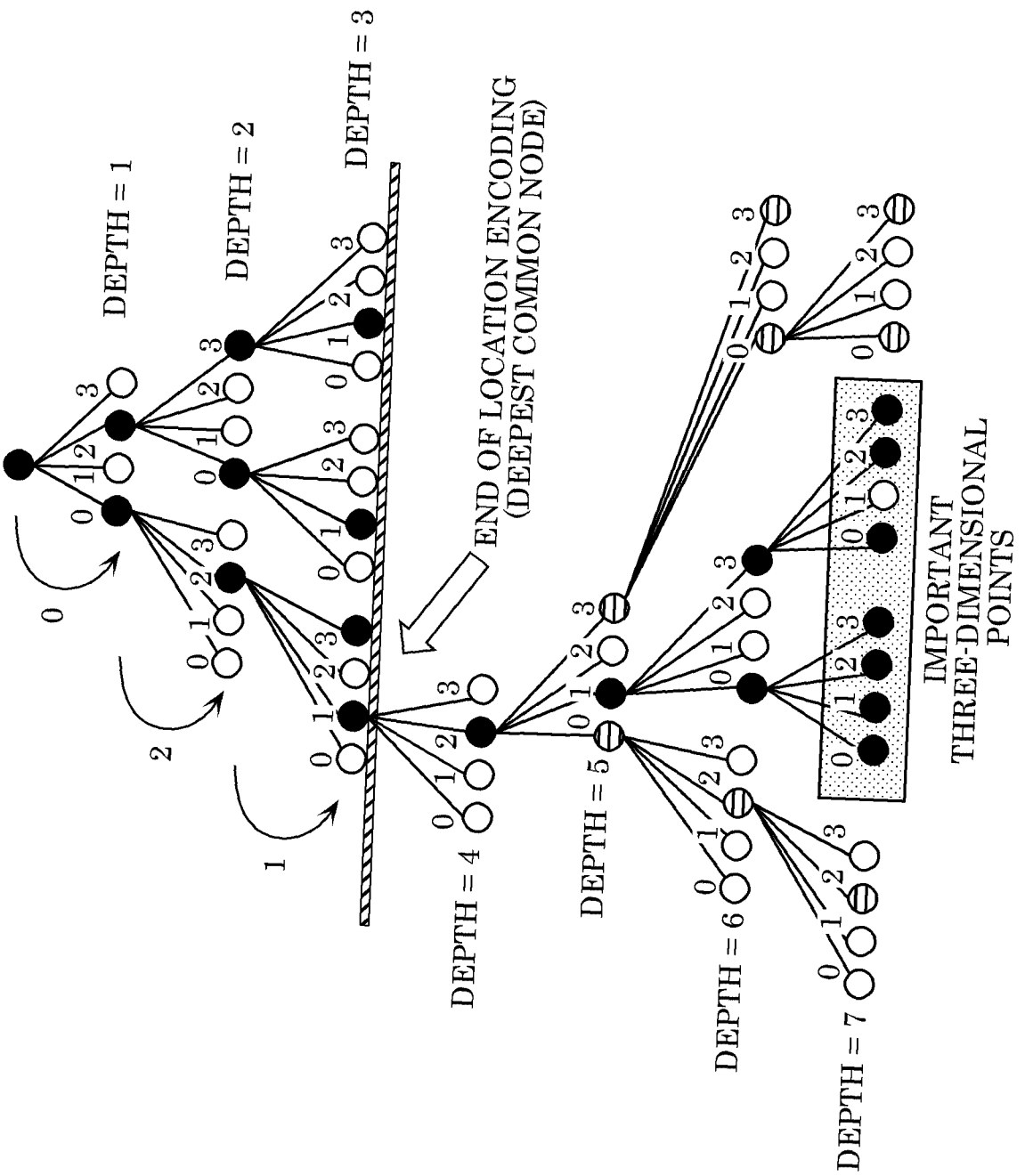

DEPTH = 1
DEPTH = 2
DEPTH = 3
END OF LOCATION ENCODING
(DEEPEST COMMON NODE)
DEPTH = 4
DEPTH = 5
DEPTH = 6
DEPTH = 7
IMPORTANT
THREE-DIMENSIONAL
POINTS

FIG. 80

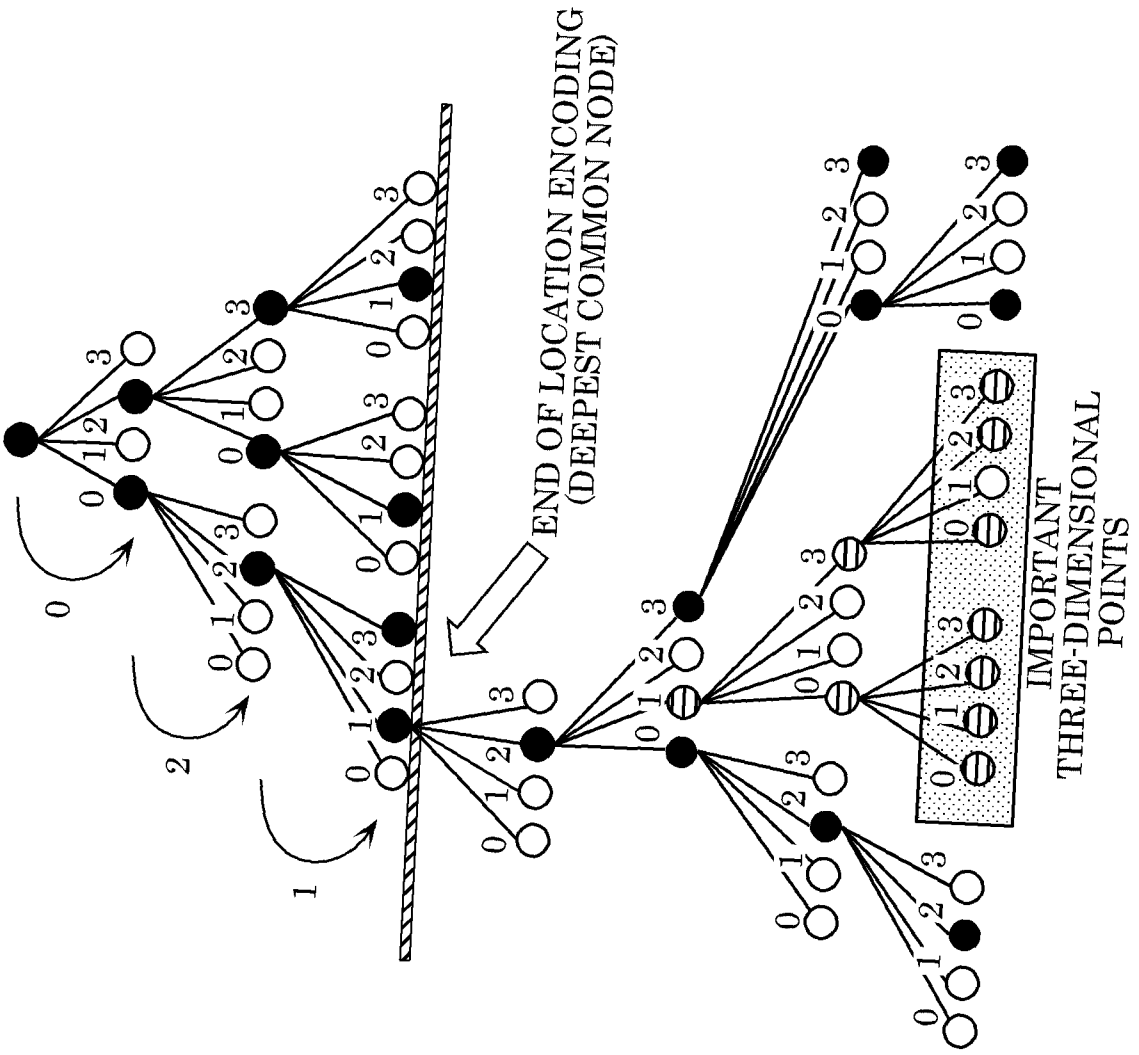

END OF LOCATION ENCODING
(DEEPEST COMMON NODE)
IMPORTANT
THREE-DIMENSIONAL
POINTS
0
1
2

FIG. 81

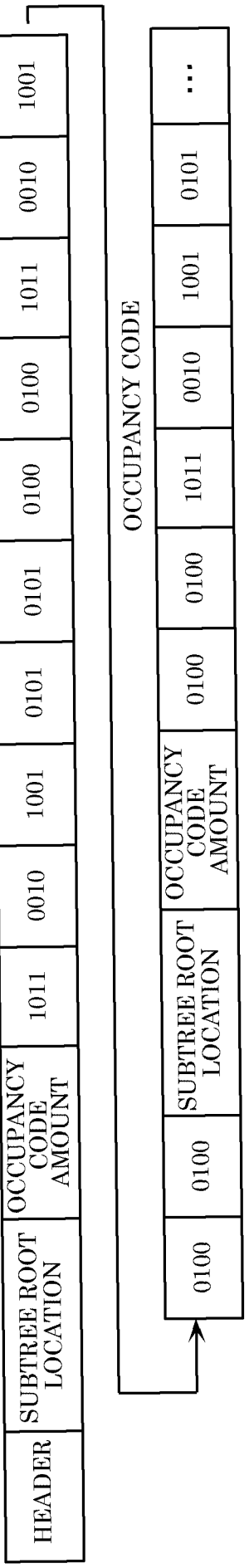

HEADER
SUBTREE ROOT LOCATION
OCCUPANCY CODE AMOUNT
1011
0010
1001
0101
0101
0100
0100
0100
1011
0010
1001
OCCUPANCY CODE
0100
0100
SUBTREE ROOT LOCATION
OCCUPANCY CODE AMOUNT
0100
0100
1011
0010
1001
0101
...
OCCUPANCY CODE

FIG. 82

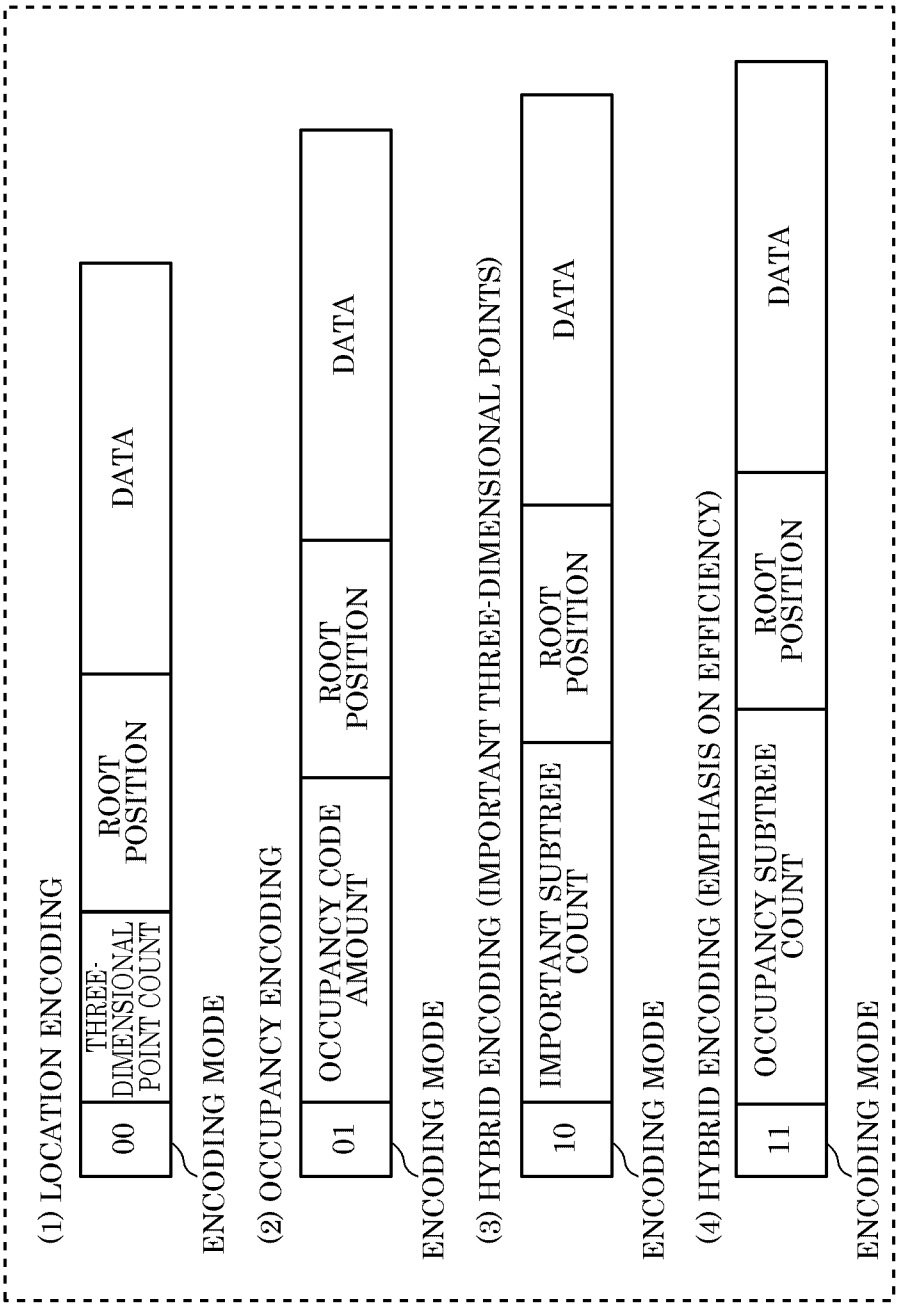

(1) LOCATION ENCODING
00
THREE-DIMENSIONAL POINT COUNT
ROOT POSITION
DATA
ENCODING MODE
(2) OCCUPANCY ENCODING
01
OCCUPANCY CODE AMOUNT
ROOT POSITION
DATA
ENCODING MODE
(3) HYBRID ENCODING (IMPORTANT THREE-DIMENSIONAL POINTS)
10
IMPORTANT SUBTREE COUNT
ROOT POSITION
DATA
ENCODING MODE
(4) HYBRID ENCODING (EMPHASIS ON EFFICIENCY)
11
OCCUPANCY SUBTREE COUNT
ROOT POSITION
DATA
ENCODING MODE

FIG. 83

```
node(depth, index) {
• • •
isleaf
if (isleaf) {   //leaf
  point_flag
}
else {          //node
 coding_type
 if (coding_type == location coding)
 {
   numPoint
   for (i=0; i<numPoint; i++) {
    num_idx[i]
    for (j=0; j<num_idx[i]; j++) {
     idx[i][j]
    }
   }
 }
 else if (coding_type==occupancy coding) {
 {
   occupancy_code
   for (i=0; i<8; i++) {
     if (occupancy_code&(0x01<<i) == 1) {
      node(depth+1, i)
     }
   }
 }
 • • •
}
}
```


CA 03090919 2020-08-10

FIG. 84

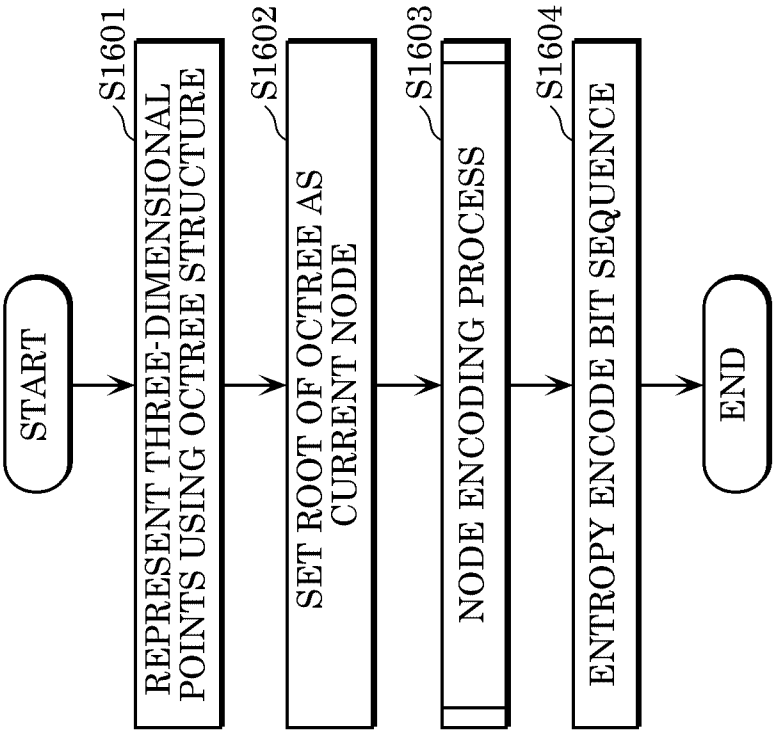

START
S1601
REPRESENT THREE-DIMENSIONAL POINTS USING OCTREE STRUCTURE
S1602
SET ROOT OF OCTREE AS CURRENT NODE
S1603
NODE ENCODING PROCESS
S1604
ENTROPY ENCODE BIT SEQUENCE
END

FIG. 85

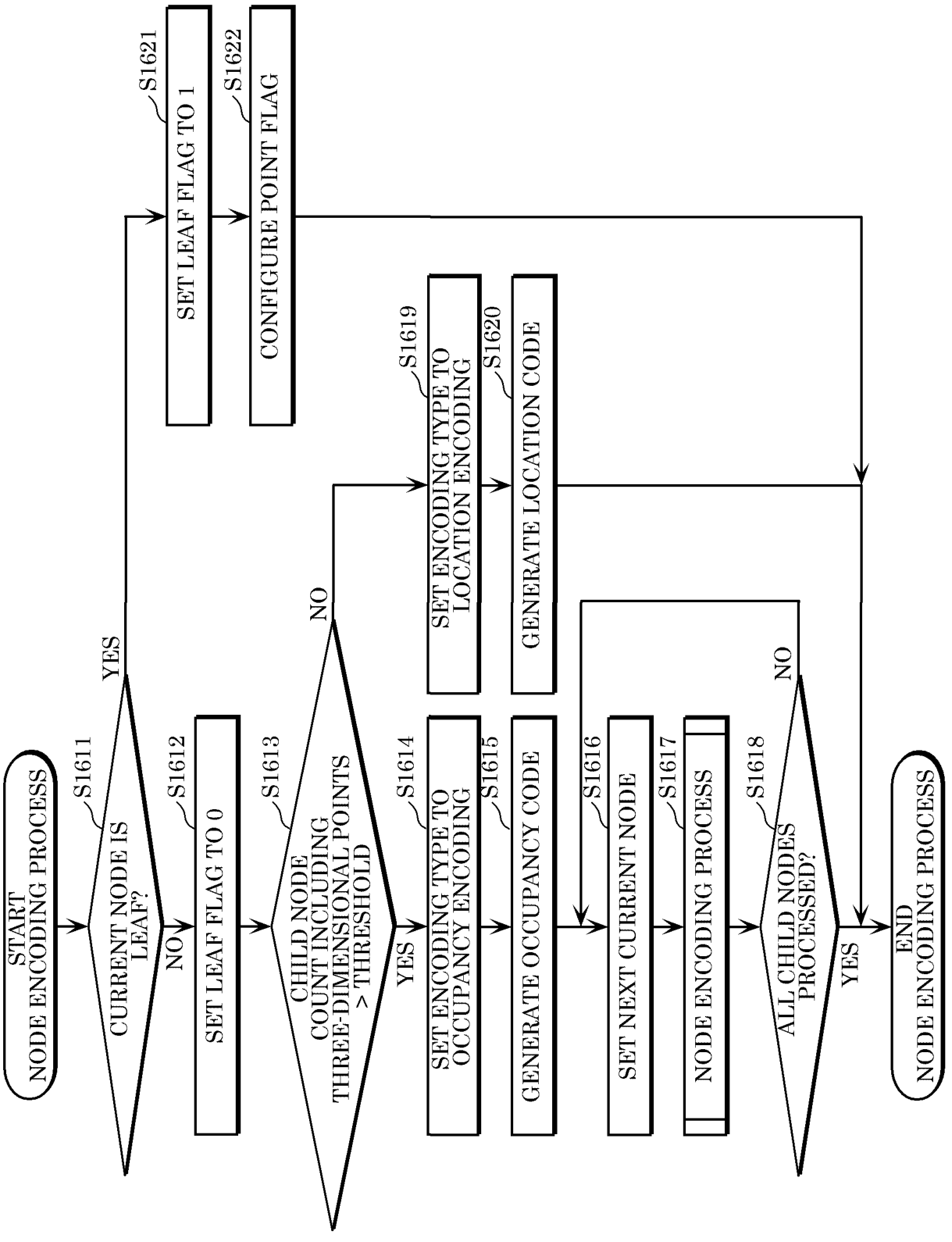

START
NODE ENCODING PROCESS
S1611
CURRENT NODE IS LEAF?
YES
NO
S1621
SET LEAF FLAG TO 1
S1622
CONFIGURE POINT FLAG
S1612
SET LEAF FLAG TO 0
S1613
CHILD NODE COUNT INCLUDING THREE-DIMENSIONAL POINTS > THRESHOLD
NO
YES
S1619
SET ENCODING TYPE TO LOCATION ENCODING
S1620
GENERATE LOCATION CODE
S1614
SET ENCODING TYPE TO OCCUPANCY ENCODING
S1615
GENERATE OCCUPANCY CODE
S1616
SET NEXT CURRENT NODE
S1617
NODE ENCODING PROCESS
S1618
ALL CHILD NODES PROCESSED?
NO
YES
END
NODE ENCODING PROCESS

FIG. 86

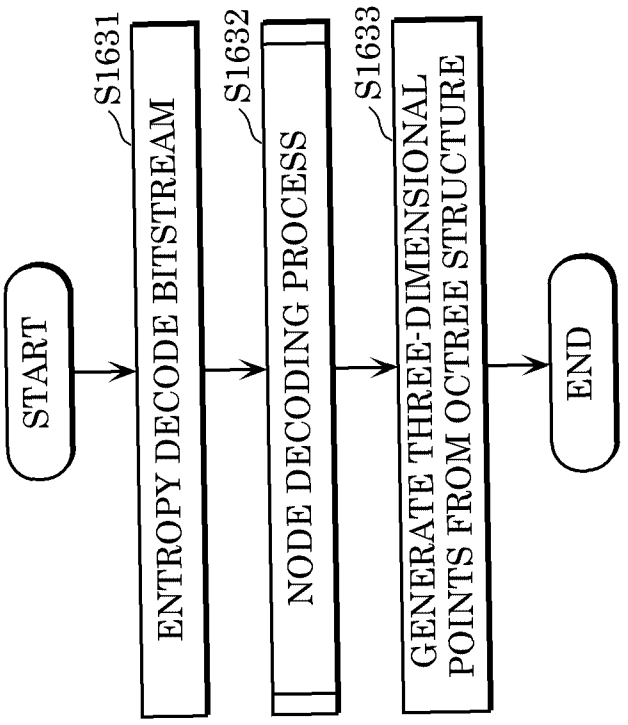

START
S1631
ENTROPY DECODE BITSTREAM
S1632
NODE DECODING PROCESS
S1633
GENERATE THREE-DIMENSIONAL POINTS FROM OCTREE STRUCTURE
END

FIG. 87

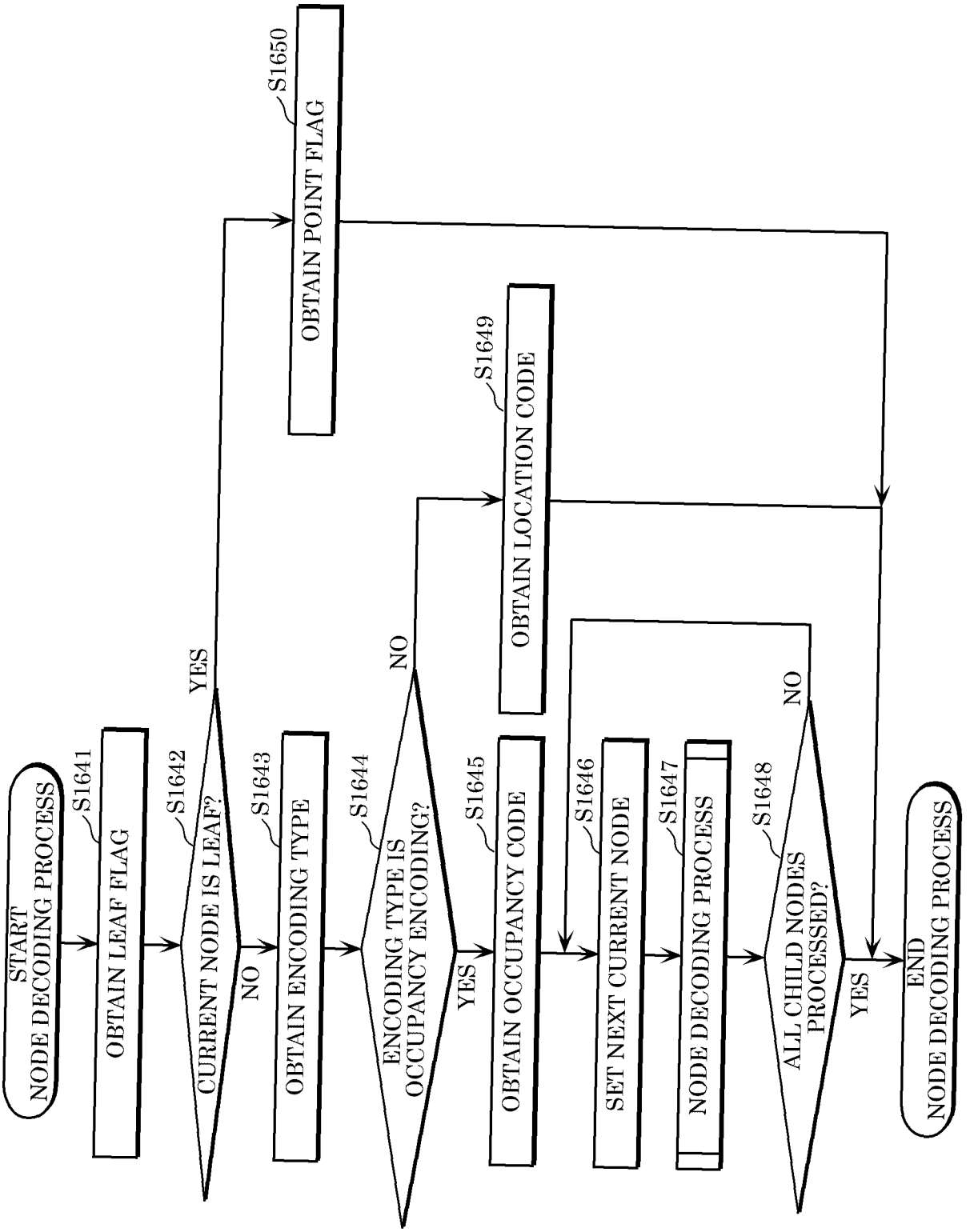

START
NODE DECODING PROCESS
S1641
OBTAIN LEAF FLAG
S1642
CURRENT NODE IS LEAF?
YES
S1650
OBTAIN POINT FLAG
NO
S1643
OBTAIN ENCODING TYPE
S1644
ENCODING TYPE IS
OCCUPANCY ENCODING?
NO
S1649
OBTAIN LOCATION CODE
YES
S1645
OBTAIN OCCUPANCY CODE
S1646
SET NEXT CURRENT NODE
S1647
NODE DECODING PROCESS
S1648
ALL CHILD NODES
PROCESSED?
NO
YES
END
NODE DECODING PROCESS

FIG. 88

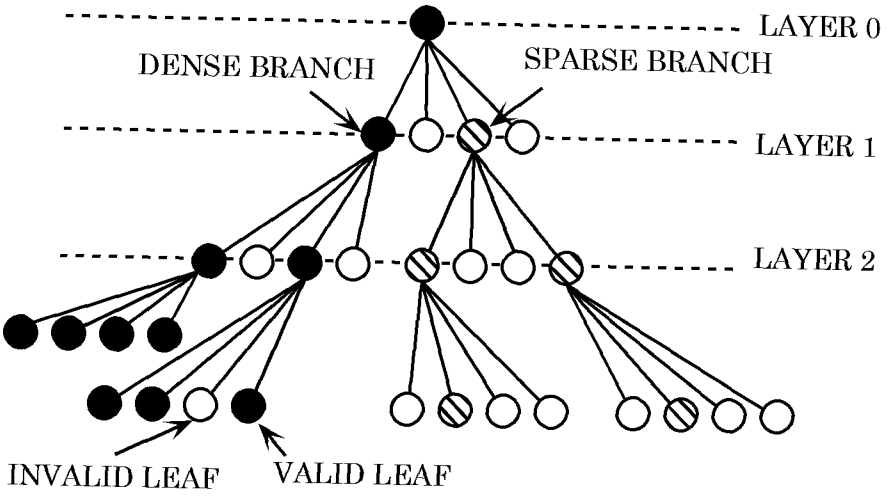

LAYER 0
DENSE BRANCH
SPARSE BRANCH
LAYER 1
LAYER 2
INVALID LEAF
VALID LEAF

FIG. 89

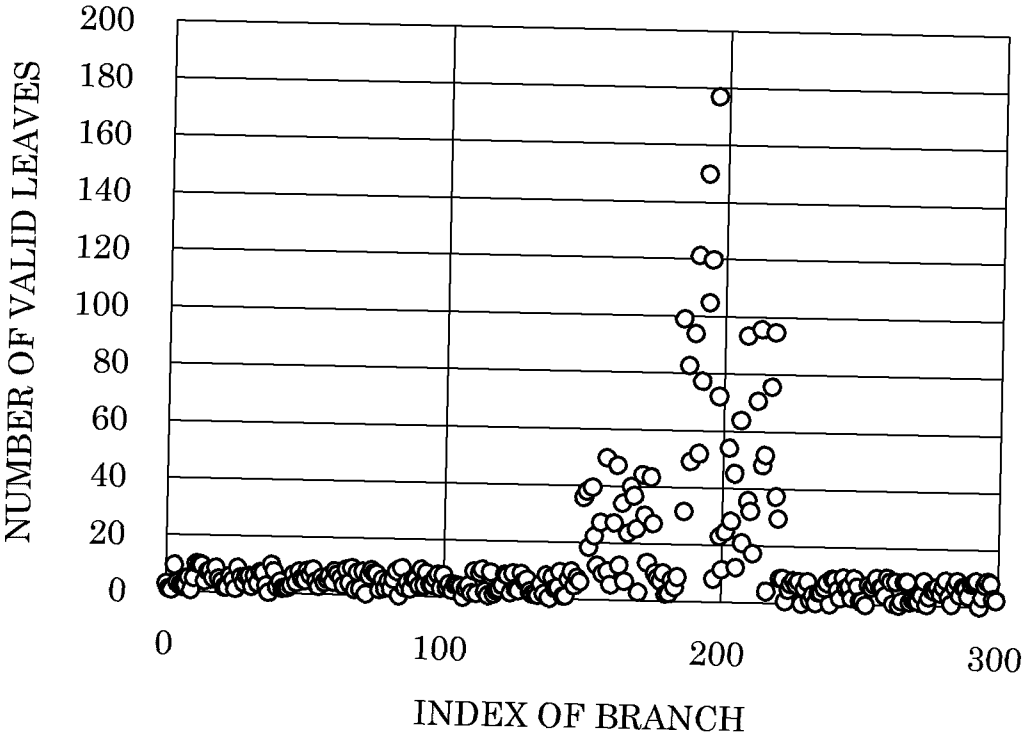

NUMBER OF VALID LEAVES
200
180
160
140
120
100
80
60
40
20
0
0
100
200
300
INDEX OF BRANCH

FIG. 90

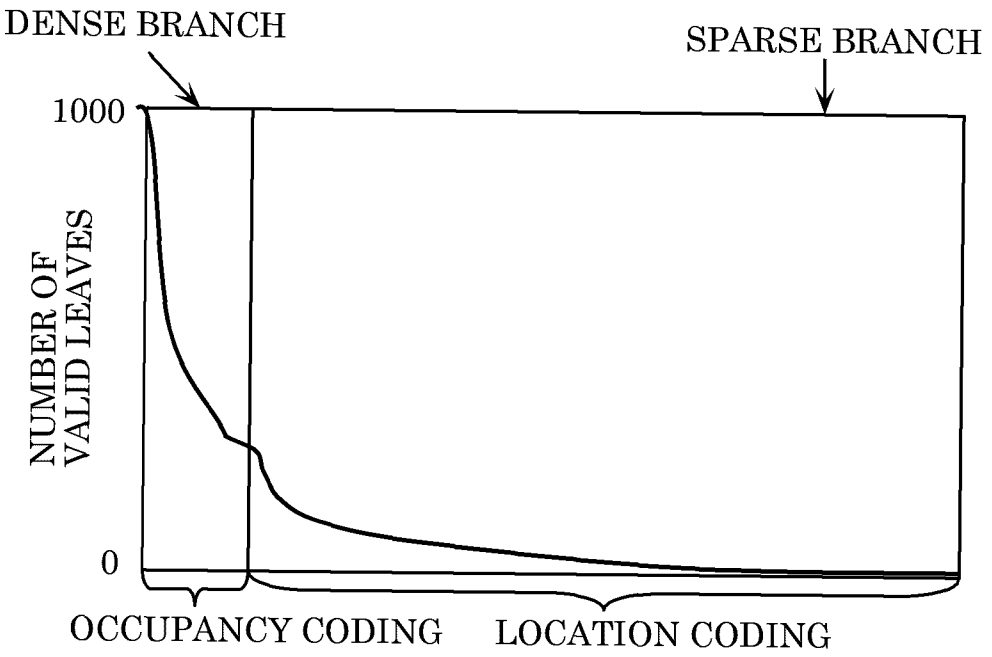

DENSE BRANCH
SPARSE BRANCH
1000
NUMBER OF VALID LEAVES
0
OCCUPANCY CODING
LOCATION CODING

FIG. 91

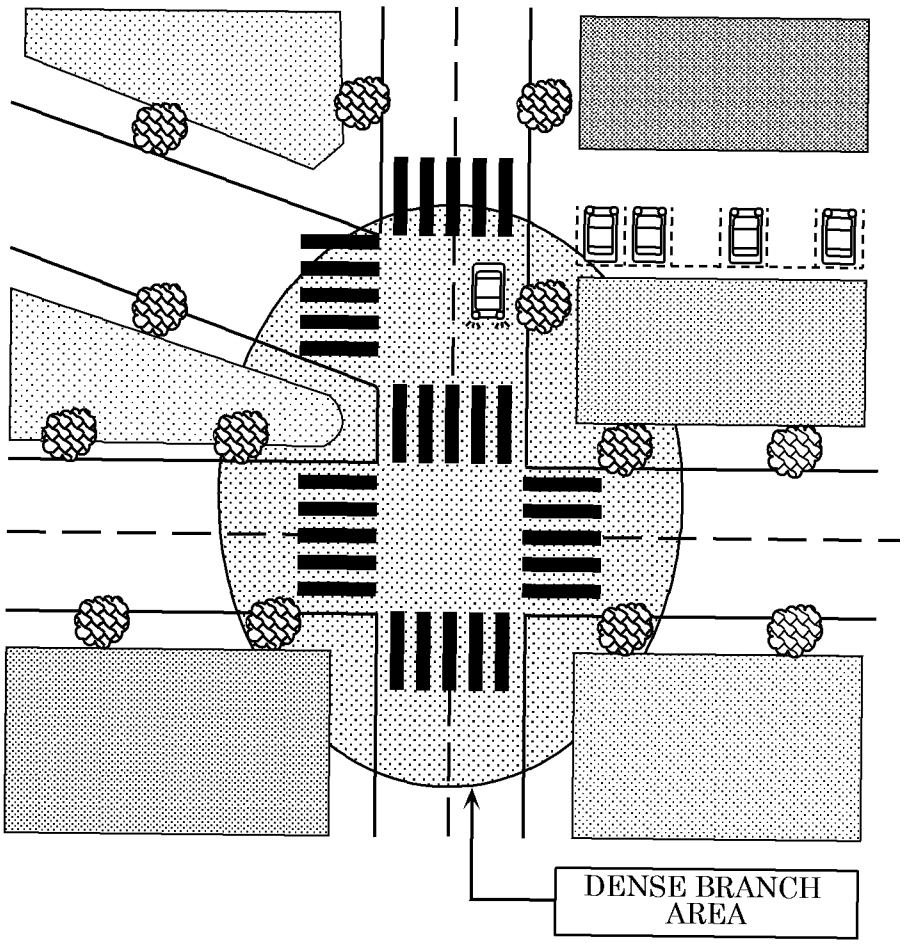

DENSE BRANCH AREA

FIG. 92

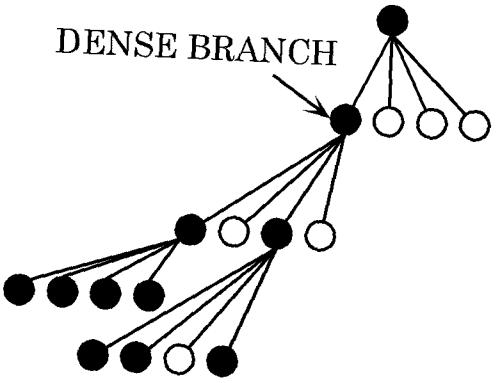


FIG. 93

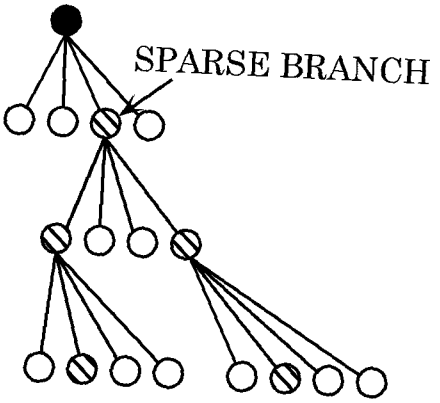


FIG. 94

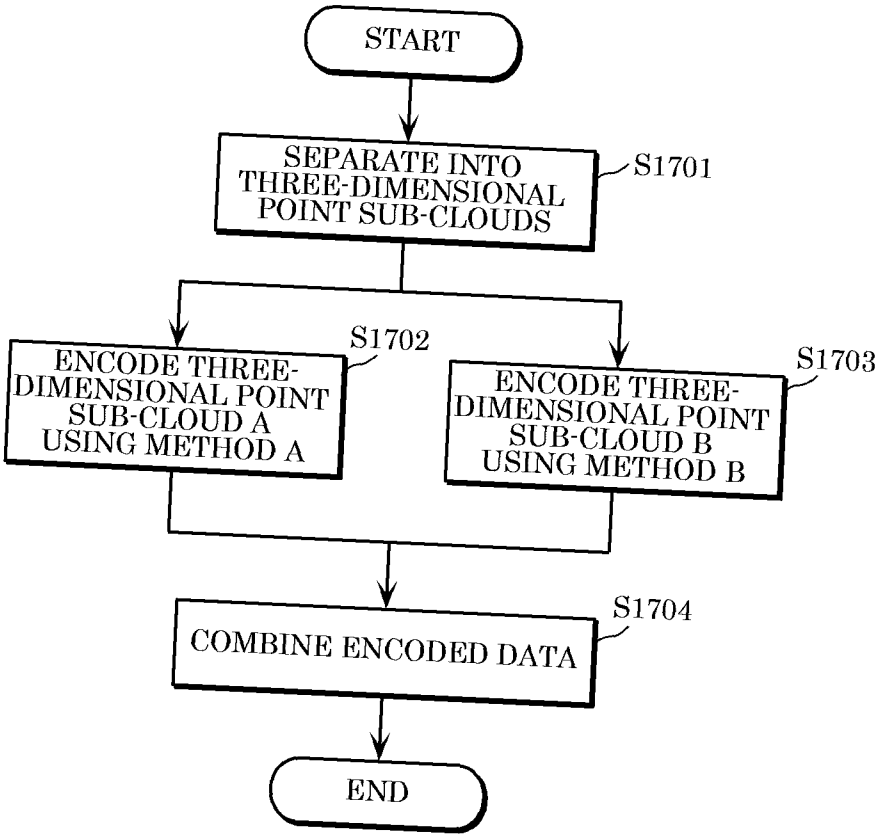


FIG. 95

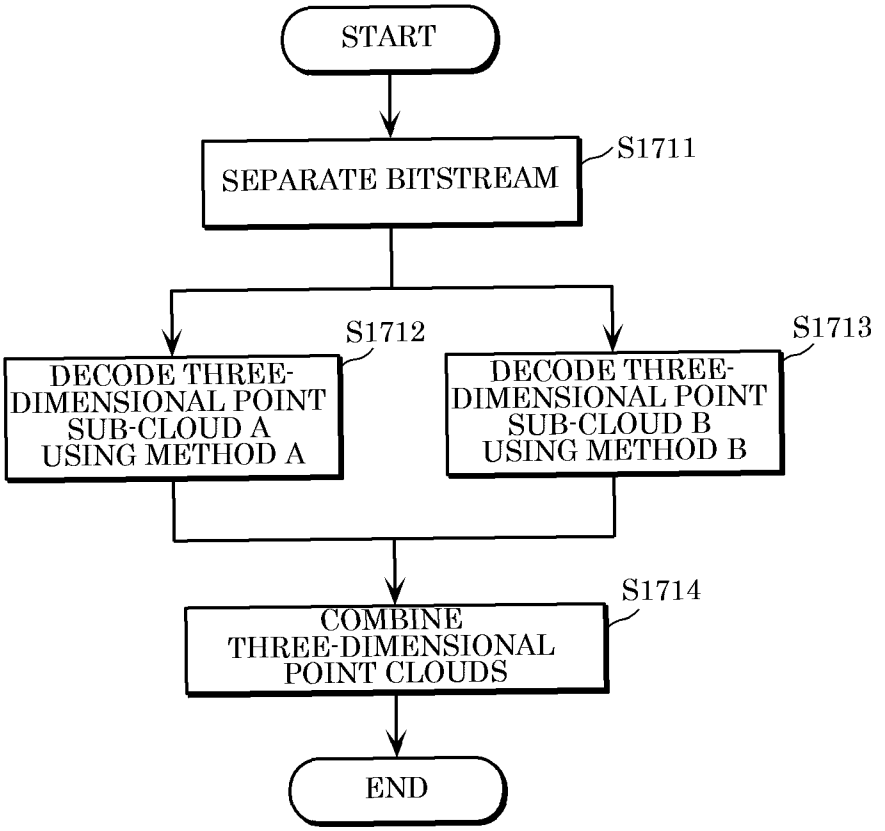

START
SEPARATE BITSTREAM
S1711
DECODE THREE-DIMENSIONAL POINT SUB-CLOUD A USING METHOD A
S1712
DECODE THREE-DIMENSIONAL POINT SUB-CLOUD B USING METHOD B
S1713
COMBINE THREE-DIMENSIONAL POINT CLOUDS
S1714
END

FIG. 96

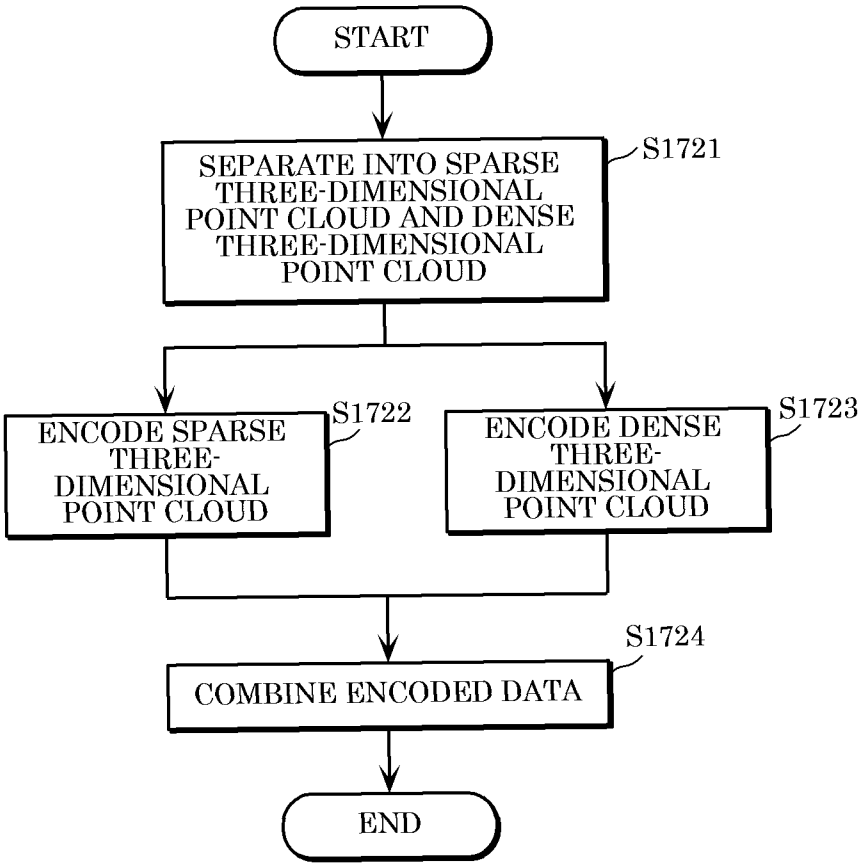

START
SEPARATE INTO SPARSE THREE-DIMENSIONAL POINT CLOUD AND DENSE THREE-DIMENSIONAL POINT CLOUD
S1721
ENCODE SPARSE THREE-DIMENSIONAL POINT CLOUD
S1722
ENCODE DENSE THREE-DIMENSIONAL POINT CLOUD
S1723
COMBINE ENCODED DATA
S1724
END

FIG. 97

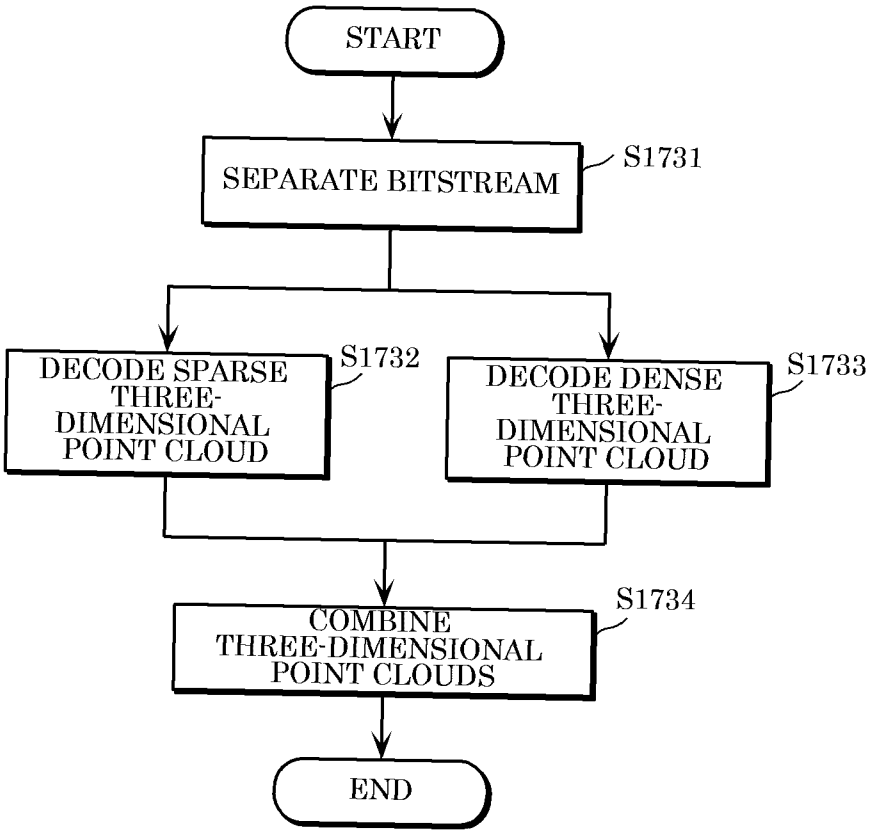

START
SEPARATE BITSTREAM
S1731
DECODE SPARSE THREE-DIMENSIONAL POINT CLOUD
S1732
DECODE DENSE THREE-DIMENSIONAL POINT CLOUD
S1733
COMBINE THREE-DIMENSIONAL POINT CLOUDS
S1734
END

FIG. 98

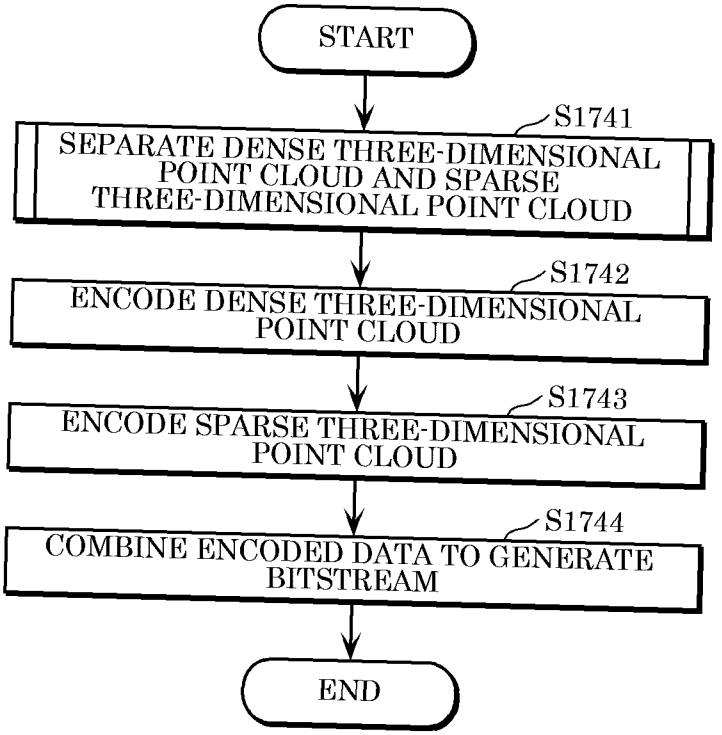

START
S1741
SEPARATE DENSE THREE-DIMENSIONAL POINT CLOUD AND SPARSE THREE-DIMENSIONAL POINT CLOUD
S1742
ENCODE DENSE THREE-DIMENSIONAL POINT CLOUD
S1743
ENCODE SPARSE THREE-DIMENSIONAL POINT CLOUD
S1744
COMBINE ENCODED DATA TO GENERATE BITSTREAM
END

FIG. 99

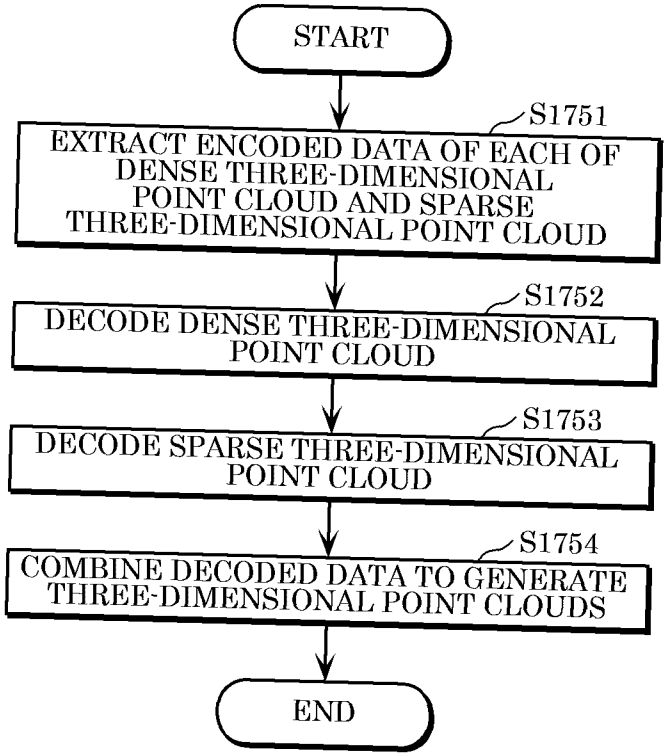

START
S1751
EXTRACT ENCODED DATA OF EACH OF DENSE THREE-DIMENSIONAL POINT CLOUD AND SPARSE THREE-DIMENSIONAL POINT CLOUD
S1752
DECODE DENSE THREE-DIMENSIONAL POINT CLOUD
S1753
DECODE SPARSE THREE-DIMENSIONAL POINT CLOUD
S1754
COMBINE DECODED DATA TO GENERATE THREE-DIMENSIONAL POINT CLOUDS
END

FIG. 100

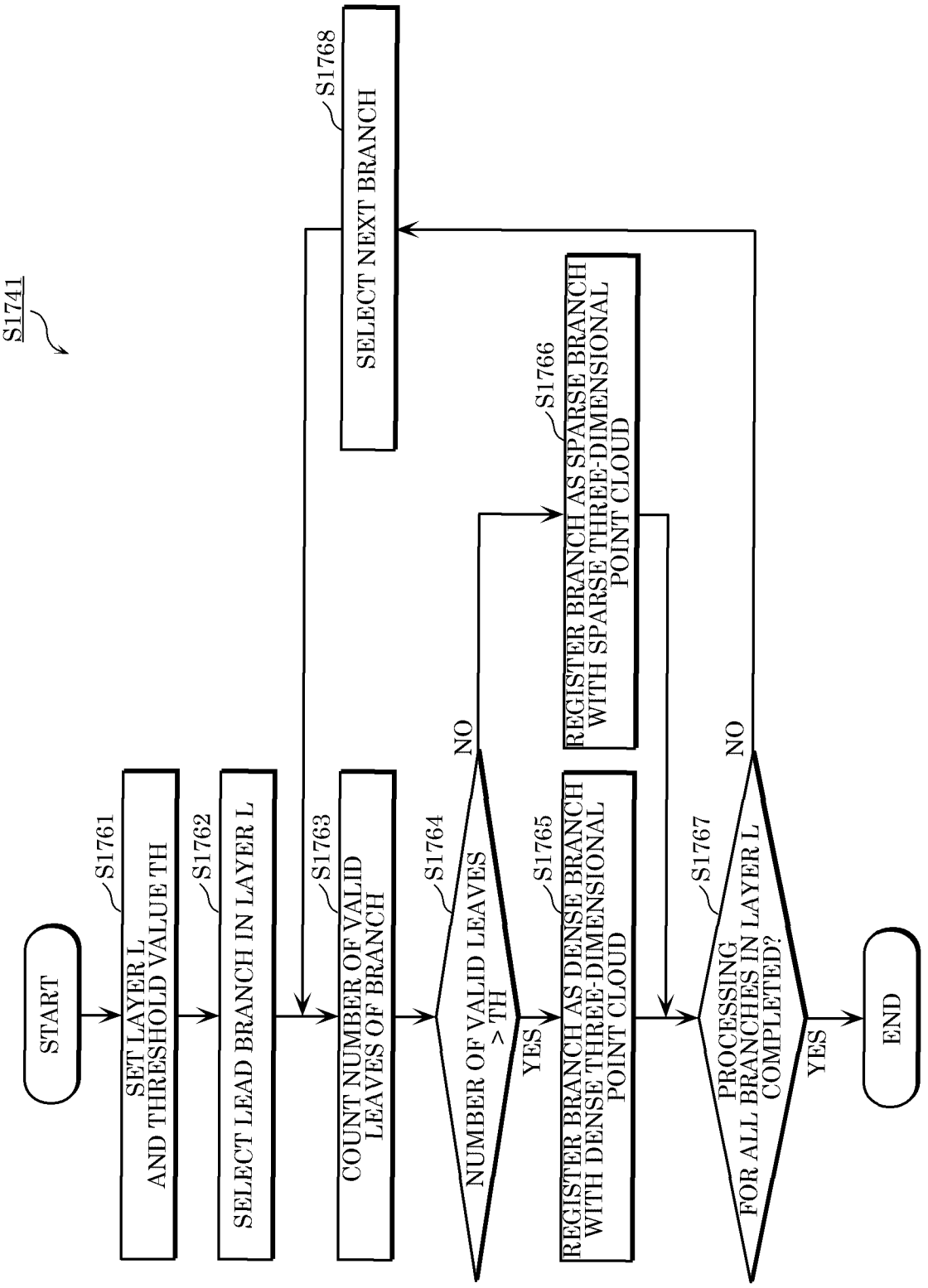

S1741
START
S1761
SET LAYER L AND THRESHOLD VALUE TH
S1762
SELECT LEAD BRANCH IN LAYER L
S1763
COUNT NUMBER OF VALID LEAVES OF BRANCH
S1764
NUMBER OF VALID LEAVES > TH
NO
YES
S1765
REGISTER BRANCH AS DENSE BRANCH WITH DENSE THREE-DIMENSIONAL POINT CLOUD
S1766
REGISTER BRANCH AS SPARSE BRANCH WITH SPARSE THREE-DIMENSIONAL POINT CLOUD
S1767
PROCESSING FOR ALL BRANCHES IN LAYER L COMPLETED?
NO
YES
S1768
SELECT NEXT BRANCH
END

FIG. 101

```
pc_header() {
 • • •
num_sub_pc
for (i=0; i<num_sub_pc; i++)
   numPoint[i]
  coding_type[i]
  data_sub_cloud[i]()
}

data_sub_cloud() {

 if (coding_type == 00)
 {
  coding_type_00_data
 }
 else if (coding_type == 01)
 {
  coding_type_01_data
 }
 • • •
end_of_data
}
```

FIG. 102

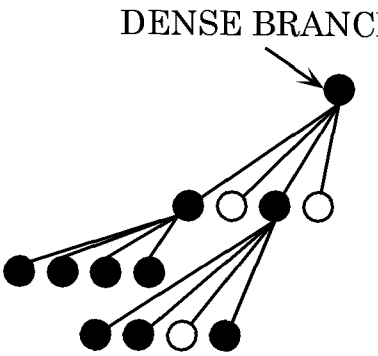


FIG. 103

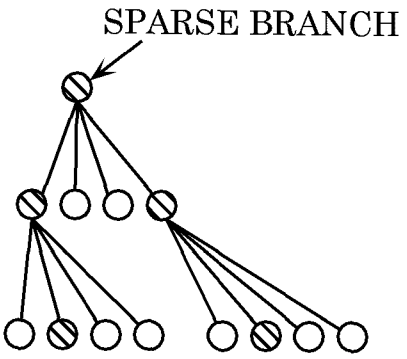


FIG. 104

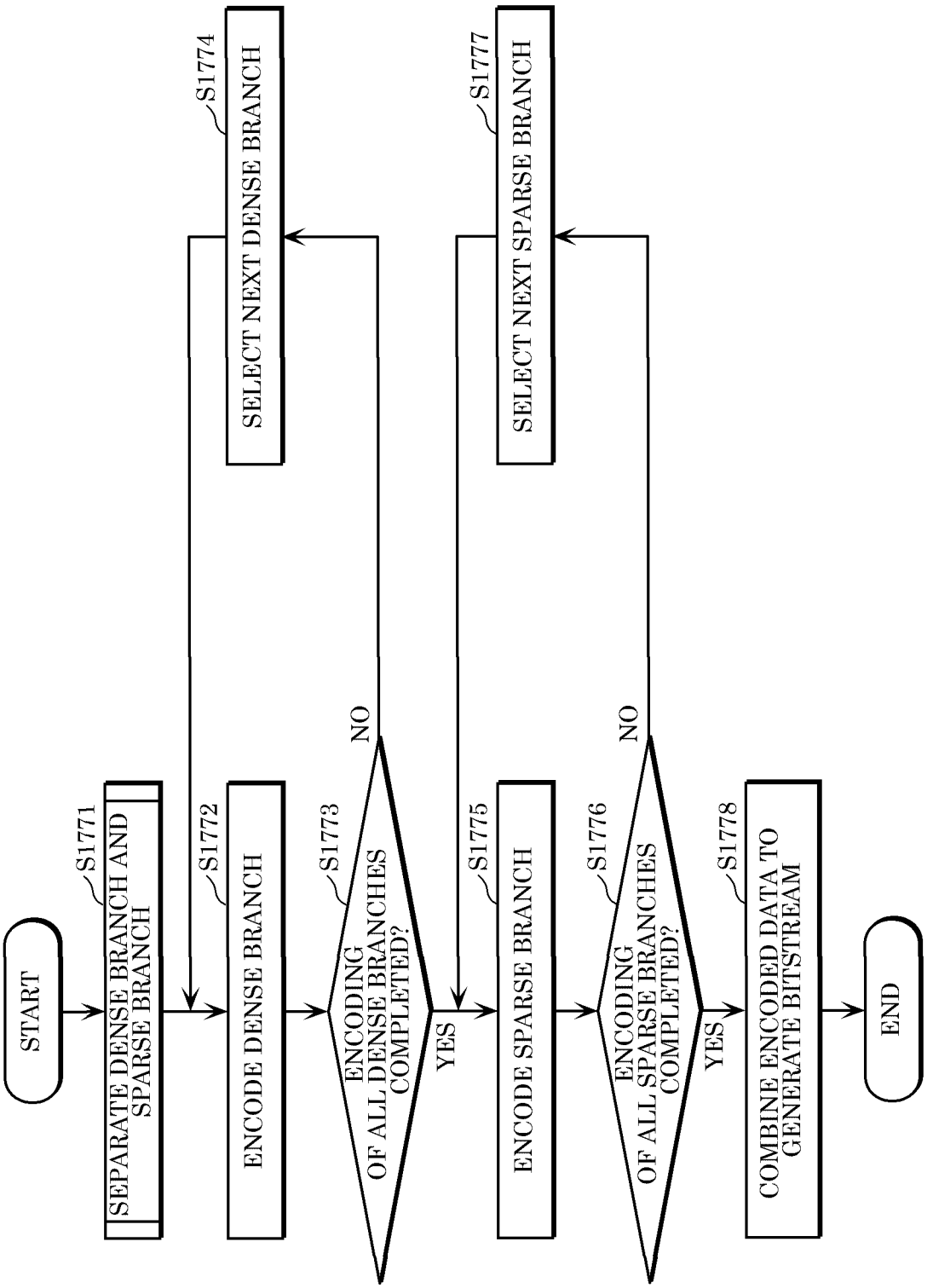

START
S1771
SEPARATE DENSE BRANCH AND SPARSE BRANCH
S1772
ENCODE DENSE BRANCH
S1773
ENCODING OF ALL DENSE BRANCHES COMPLETED?
NO
YES
S1774
SELECT NEXT DENSE BRANCH
S1775
ENCODE SPARSE BRANCH
S1776
ENCODING OF ALL SPARSE BRANCHES COMPLETED?
NO
YES
S1777
SELECT NEXT SPARSE BRANCH
S1778
COMBINE ENCODED DATA TO GENERATE BITSTREAM
END

FIG. 105

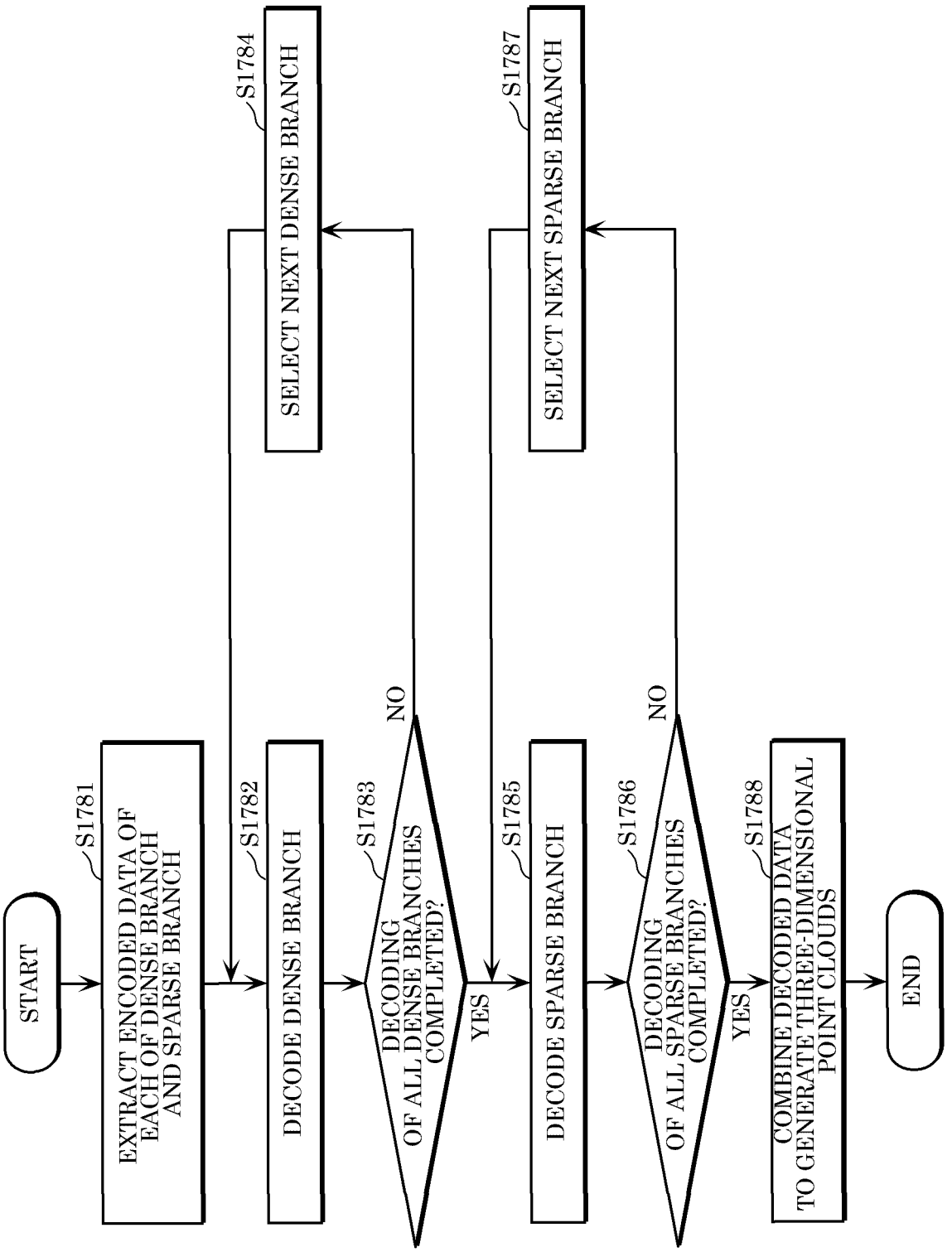

START
S1781
EXTRACT ENCODED DATA OF EACH OF DENSE BRANCH AND SPARSE BRANCH
S1782
DECODE DENSE BRANCH
S1783
DECODING OF ALL DENSE BRANCHES COMPLETED?
NO
YES
S1784
SELECT NEXT DENSE BRANCH
S1785
DECODE SPARSE BRANCH
S1786
DECODING OF ALL SPARSE BRANCHES COMPLETED?
NO
YES
S1787
SELECT NEXT SPARSE BRANCH
S1788
COMBINE DECODED DATA TO GENERATE THREE-DIMENSIONAL POINT CLOUDS
END

FIG. 106

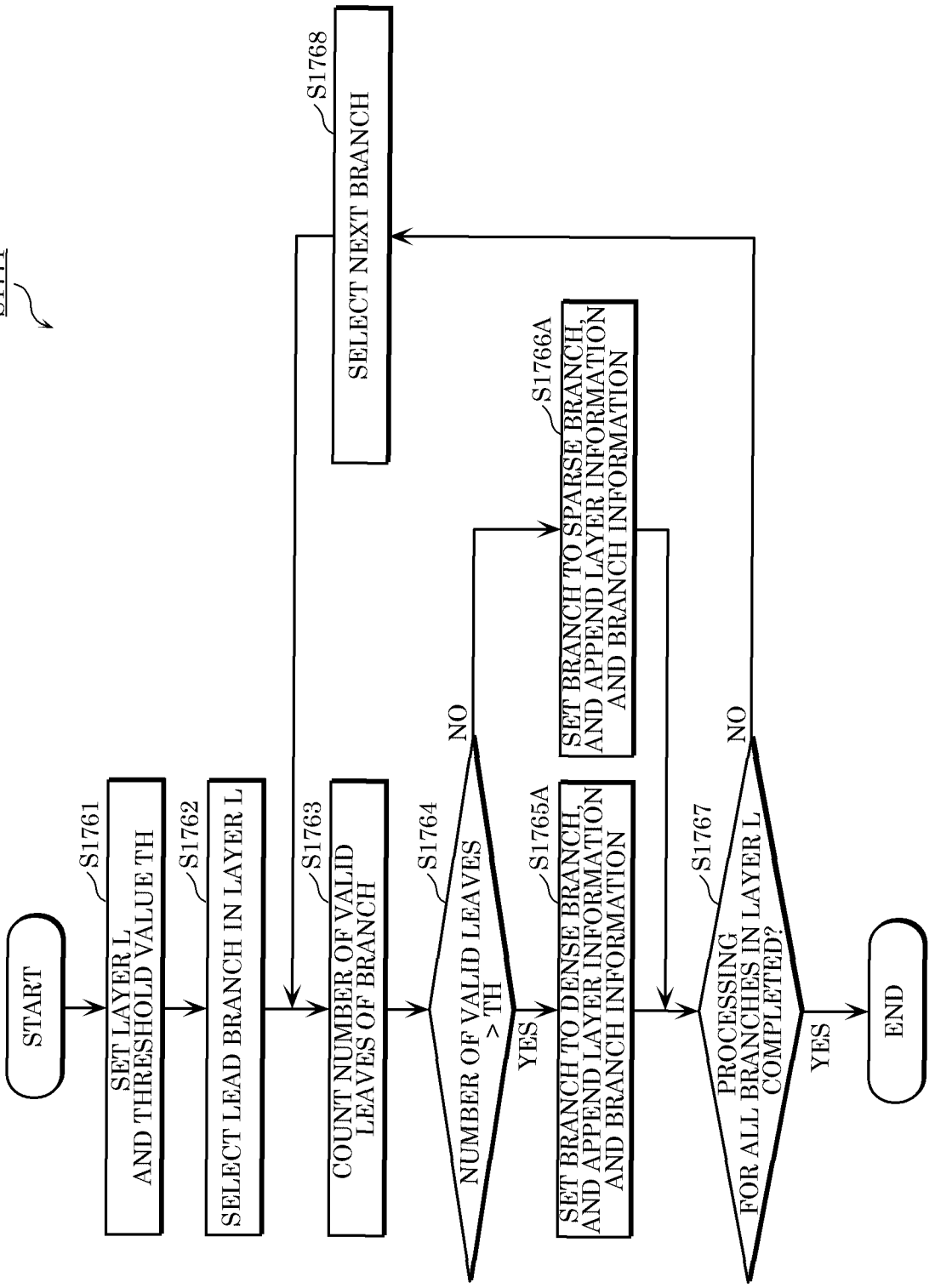

S1771
START
S1761
SET LAYER L AND THRESHOLD VALUE TH
S1762
SELECT LEAD BRANCH IN LAYER L
S1763
COUNT NUMBER OF VALID LEAVES OF BRANCH
S1764
NUMBER OF VALID LEAVES > TH
NO
YES
S1765A
SET BRANCH TO DENSE BRANCH, AND APPEND LAYER INFORMATION AND BRANCH INFORMATION
S1766A
SET BRANCH TO SPARSE BRANCH, AND APPEND LAYER INFORMATION AND BRANCH INFORMATION
S1767
PROCESSING FOR ALL BRANCHES IN LAYER L COMPLETED?
NO
YES
S1768
SELECT NEXT BRANCH
END

FIG. 107

```
pc_header() {
• • •
num_sub_pc
for (i=0; i<num_sub_pc; i++)
  layer_id[i]
  branch_id[i]
   numPoint[i]
  coding_type[i]
  data_sub_cloud[i]()
}

data_sub_cloud() {

if (coding_type == 00)
{
  coding_type_00_data
}
else if (coding_type == 01)
{
  coding_type_01_data
}
• • •
end_of_data
}
```

FIG. 108

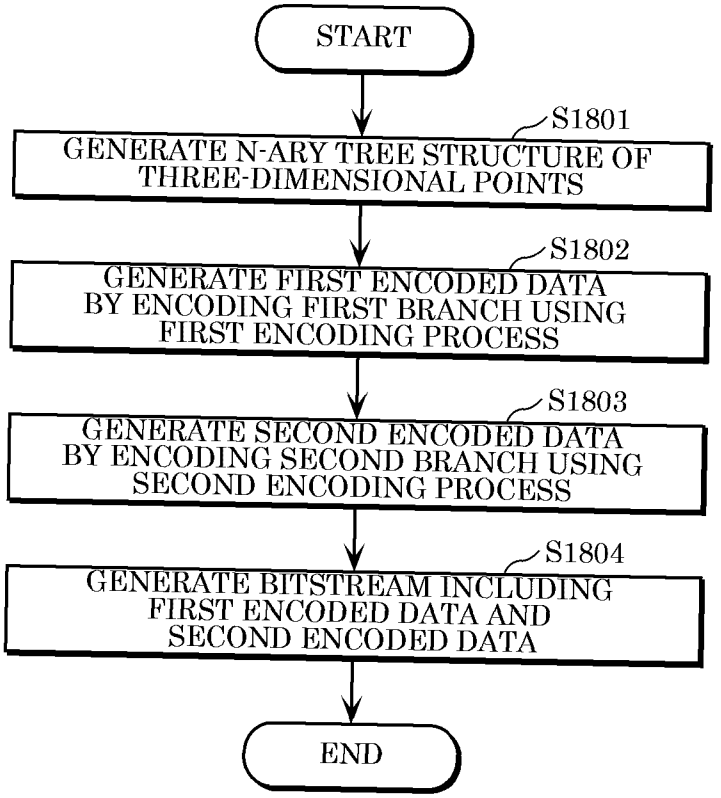


FIG. 109

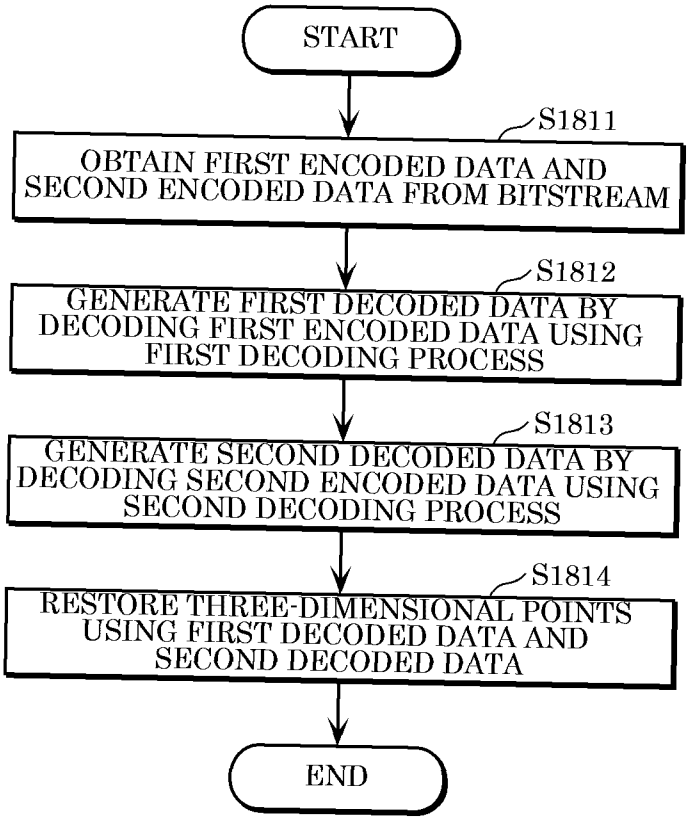


FIG. 110

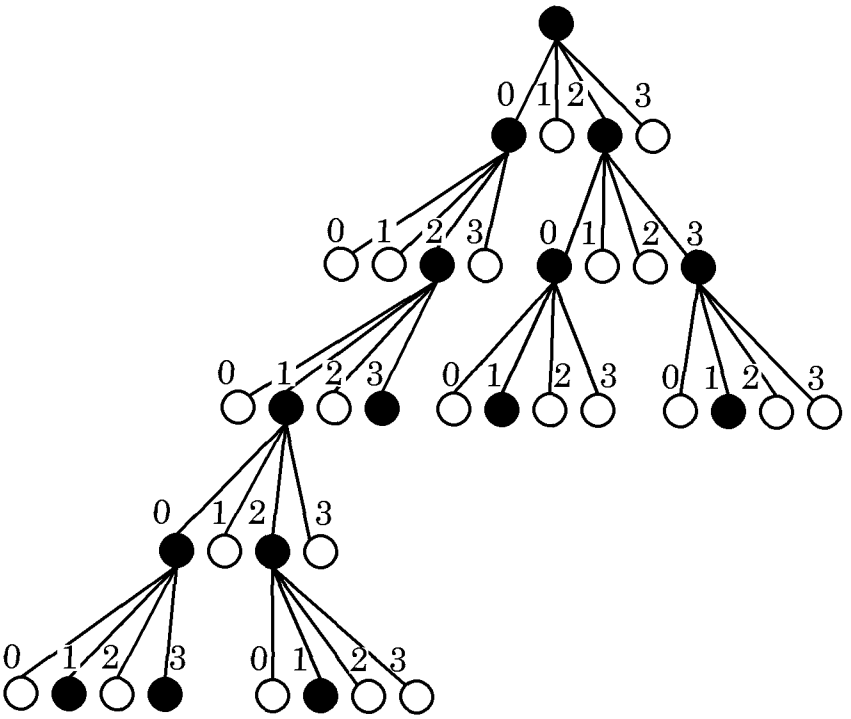


FIG. 111

OCCUPANCY CODE		
1010		
0010	1001	
0101	0100	0100
1010		
0101	0100	

FIG. 112

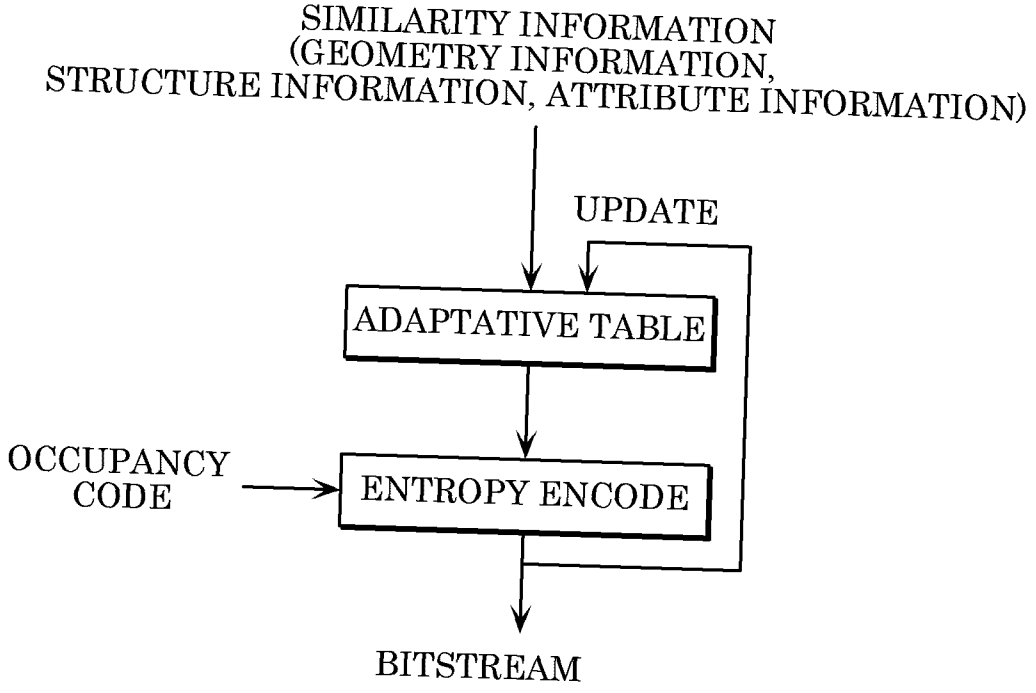


FIG. 113

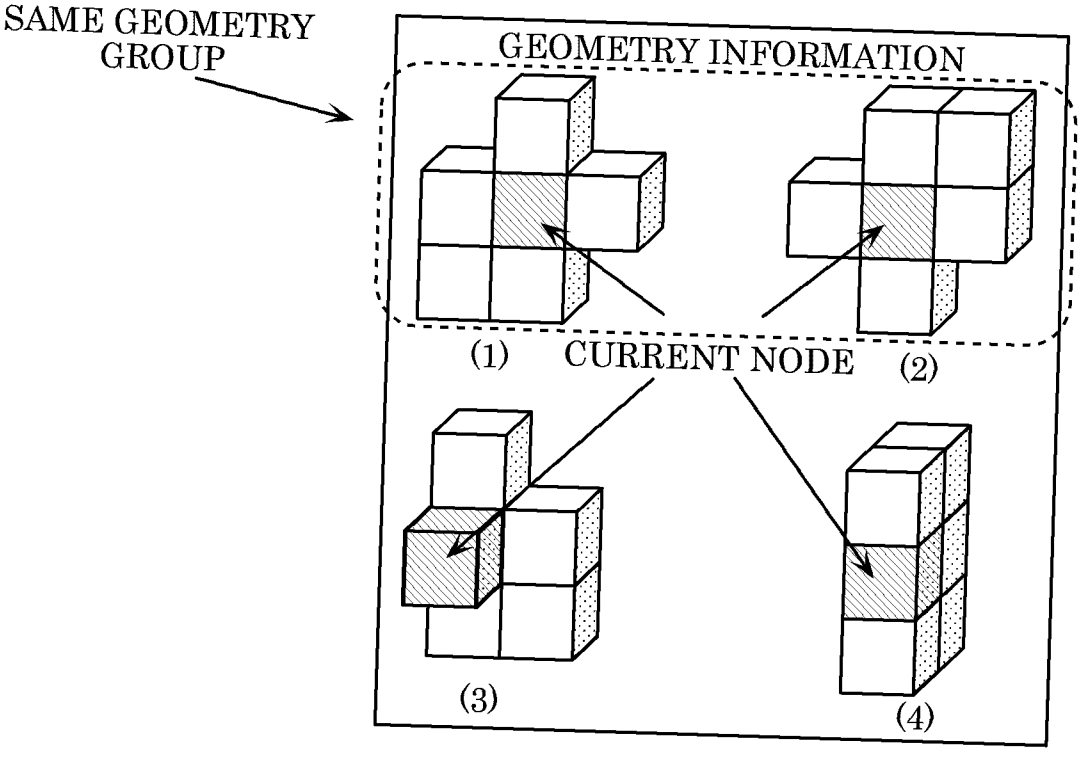


FIG. 114

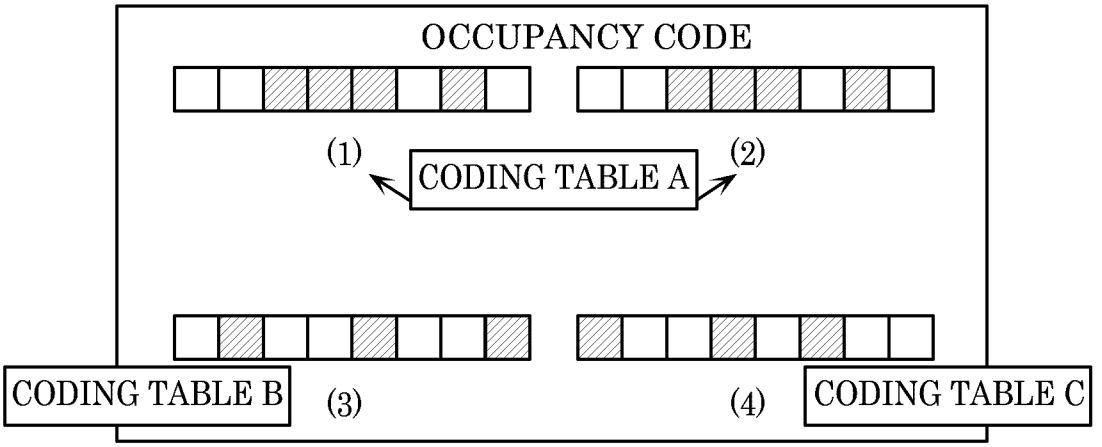

OCCUPANCY CODE
(1)
CODING TABLE A
(2)
CODING TABLE B
(3)
(4)
CODING TABLE C

FIG. 115

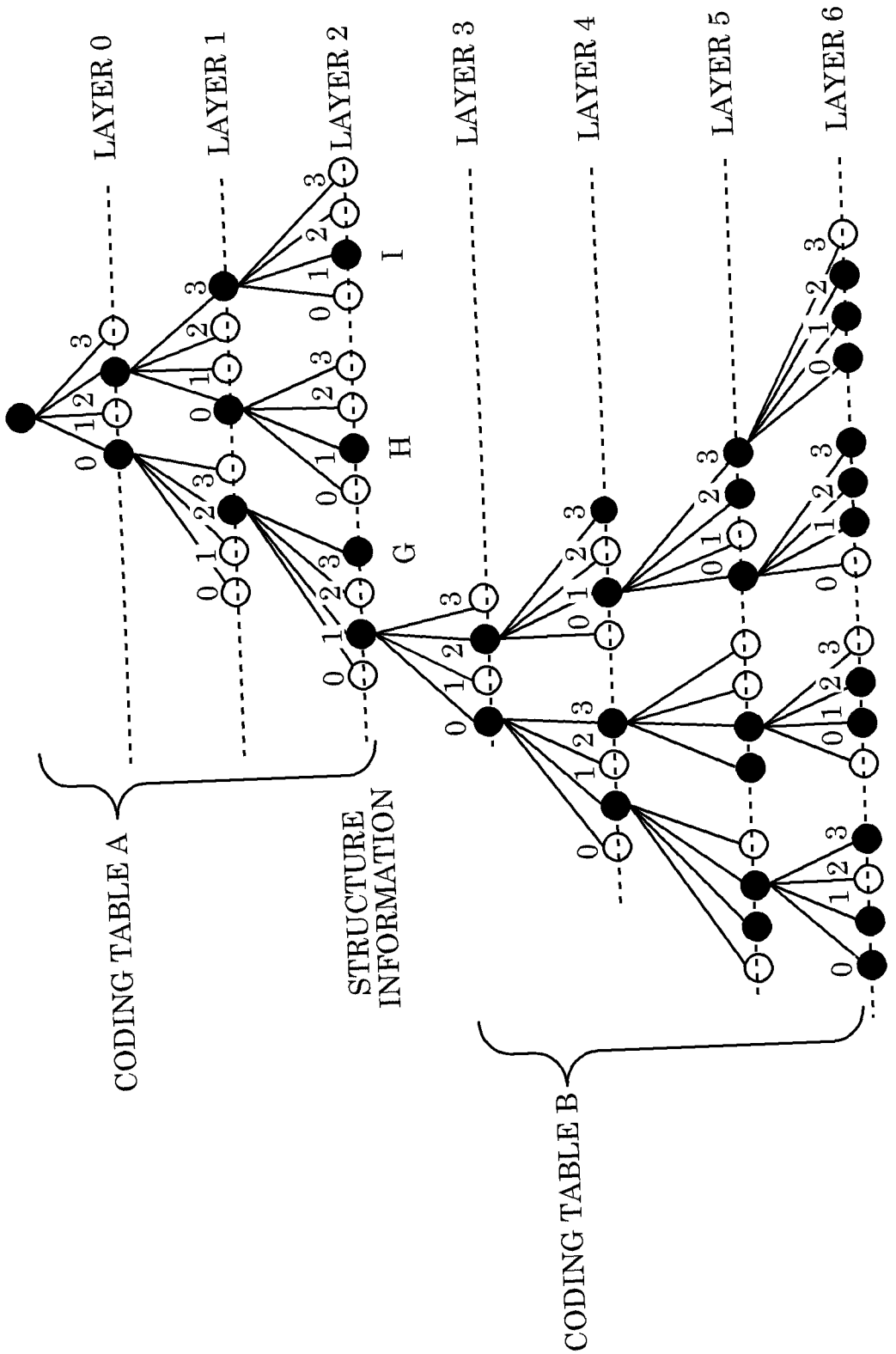

LAYER 0
LAYER 1
LAYER 2
LAYER 3
LAYER 4
LAYER 5
LAYER 6
CODING TABLE A
STRUCTURE INFORMATION
CODING TABLE B
G
H
I

FIG. 116

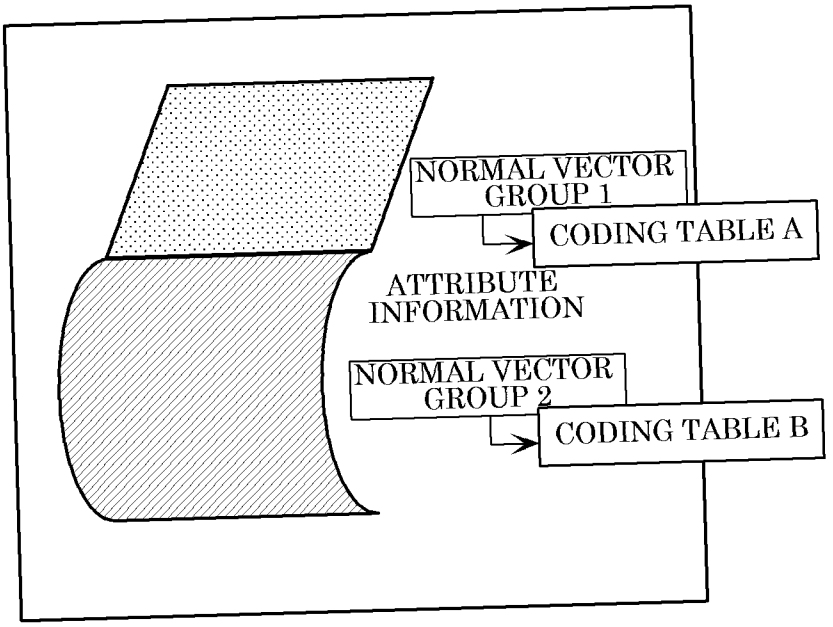


FIG. 117

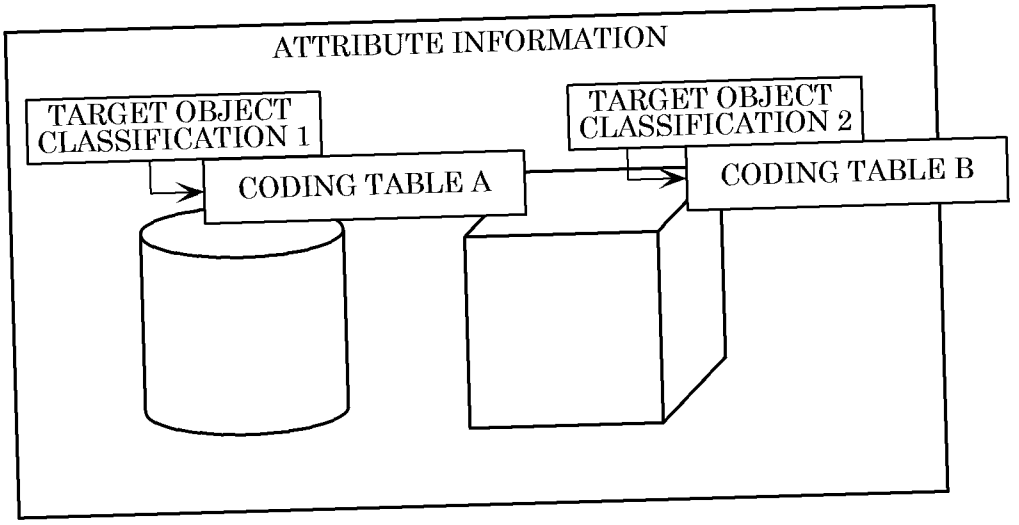


FIG. 118

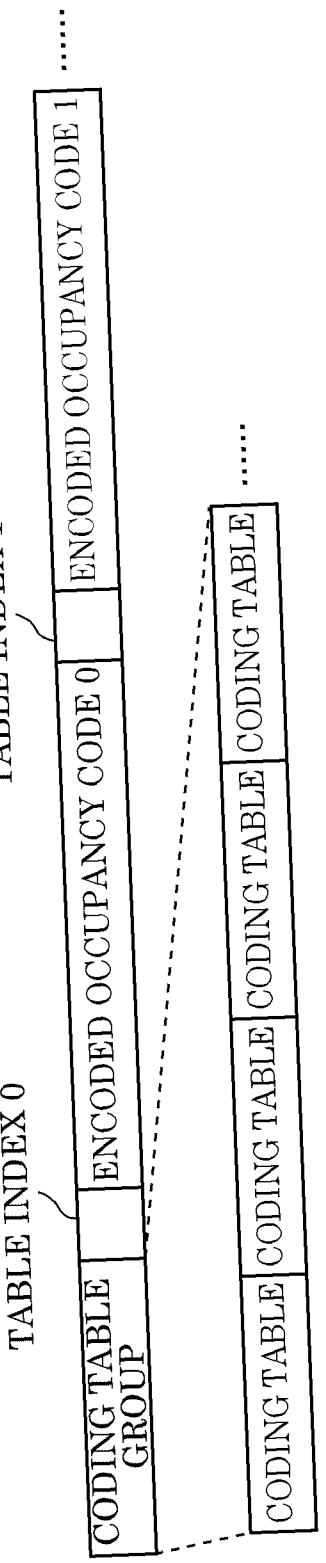

TABLE INDEX 0
TABLE INDEX 1
CODING TABLE GROUP
ENCODED OCCUPANCY CODE 0
ENCODED OCCUPANCY CODE 1
CODING TABLE
CODING TABLE
CODING TABLE
CODING TABLE
.....
.....

FIG. 119

TABLE INDEX 0

OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
00000000	MODEL 0-0	TYPE A
00000001		
.....	.....	.....
11111111	MODEL 0-X	TYPE X

FIG. 120

TABLE INDEX 1

OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
00000000	MODEL 1-0	TYPE A
00000001	MODEL 1-1	TYPE B
.....	.....	.....
11111111	MODEL 1-Y	TYPE Y

FIG. 121

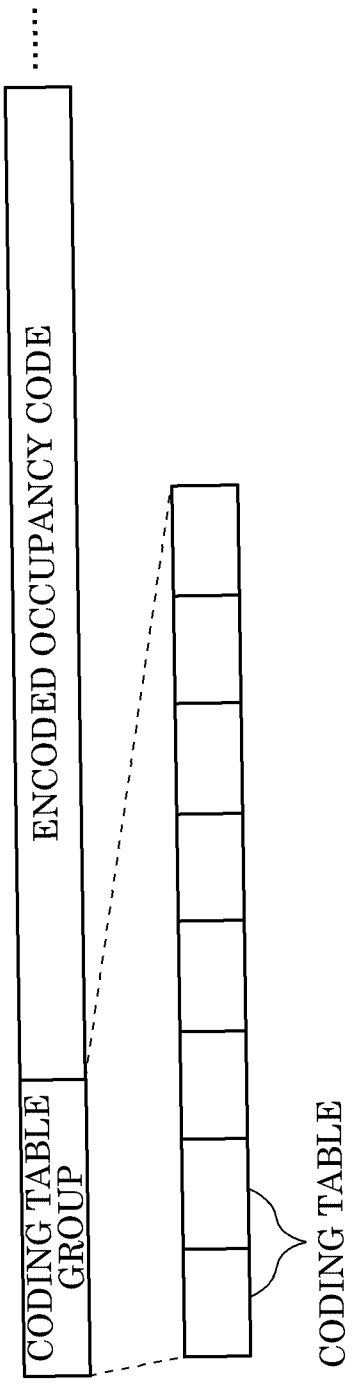


FIG. 122

TABLE INDEX 0

BIT NUMBER OF OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
1	MODEL 0-0	TYPE A
2	MODEL 0-1	
.....	.....	.....
8	MODEL 0-7	TYPE B

FIG. 123

TABLE INDEX 1

BIT NUMBER OF OCCUPANCY CODE	CONTEXT MODEL	CONTEXT MODEL TYPE
1	MODEL 1-0	TYPE A
2		TYPE A
.....	.....	.....
8	MODEL 1-Y	TYPE B

FIG. 124

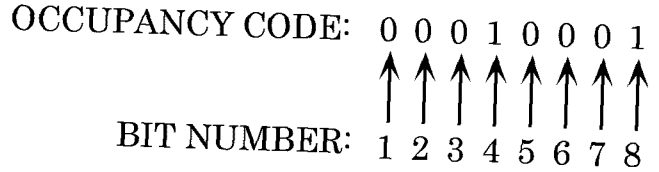


FIG. 125

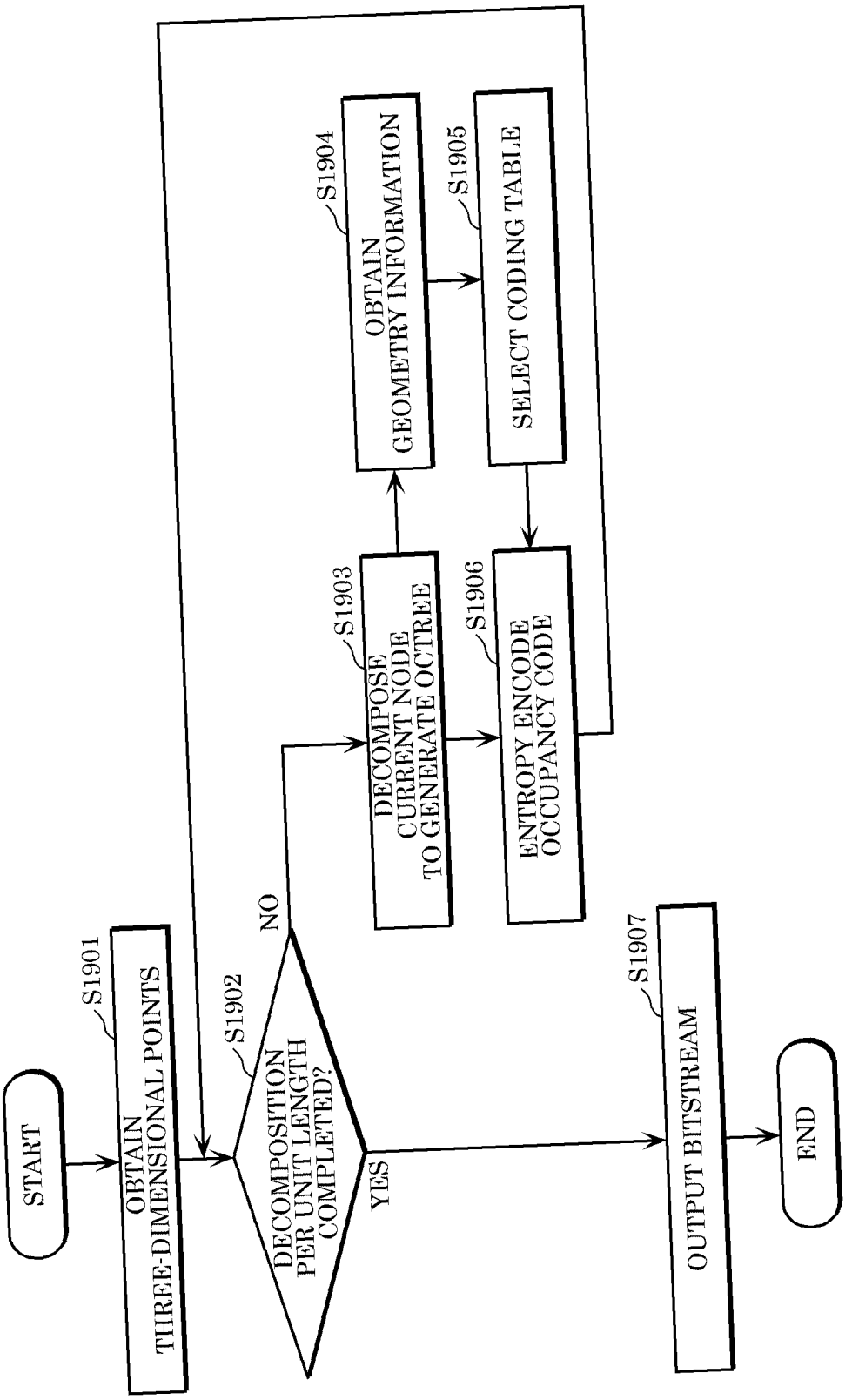

START
S1901
OBTAIN THREE-DIMENSIONAL POINTS
S1902
DECOMPOSITION PER UNIT LENGTH COMPLETED?
NO
YES
S1903
DECOMPOSE CURRENT NODE TO GENERATE OCTREE
S1904
OBTAIN GEOMETRY INFORMATION
S1905
SELECT CODING TABLE
S1906
ENTROPY ENCODE OCCUPANCY CODE
S1907
OUTPUT BITSTREAM
END

FIG. 126

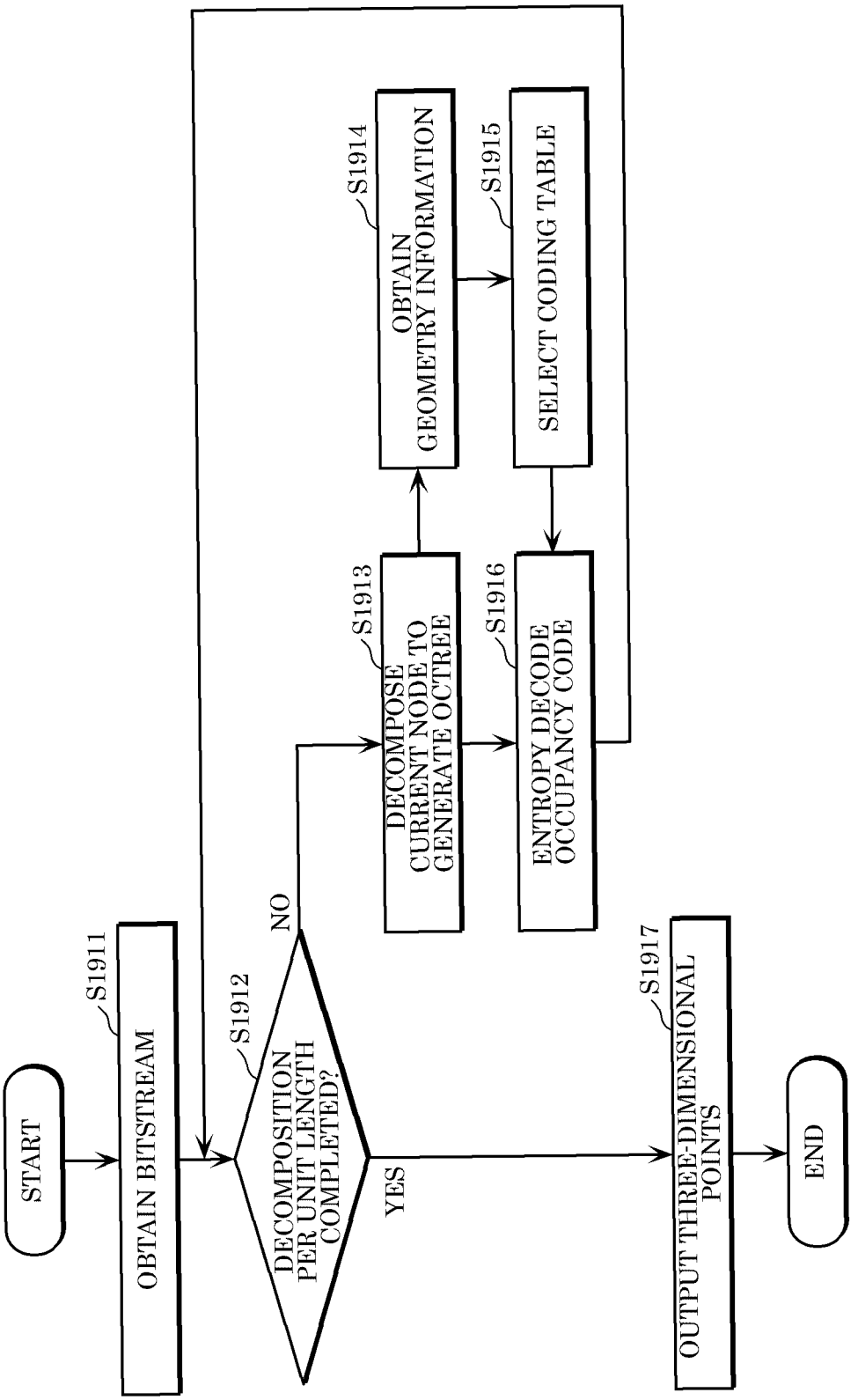

START
S1911
OBTAIN BITSTREAM
S1912
DECOMPOSITION PER UNIT LENGTH COMPLETED?
NO
YES
S1913
DECOMPOSE CURRENT NODE TO GENERATE OCTREE
S1914
OBTAIN GEOMETRY INFORMATION
S1915
SELECT CODING TABLE
S1916
ENTROPY DECODE OCCUPANCY CODE
S1917
OUTPUT THREE-DIMENSIONAL POINTS
END

FIG. 127

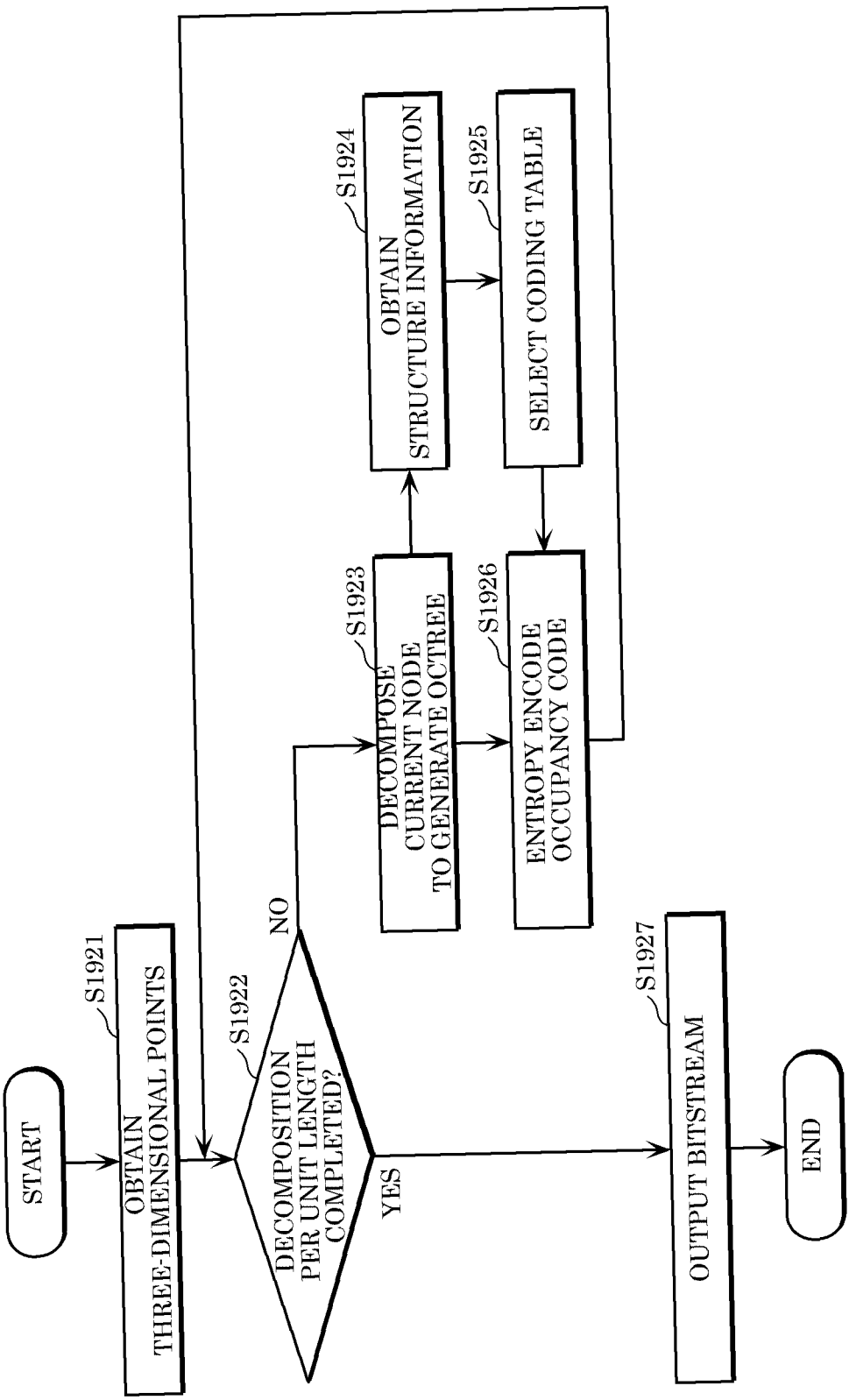

START
S1921
OBTAIN THREE-DIMENSIONAL POINTS
S1922
DECOMPOSITION PER UNIT LENGTH COMPLETED?
NO
YES
S1923
DECOMPOSE CURRENT NODE TO GENERATE OCTREE
S1924
OBTAIN STRUCTURE INFORMATION
S1925
SELECT CODING TABLE
S1926
ENTROPY ENCODE OCCUPANCY CODE
S1927
OUTPUT BITSTREAM
END

FIG. 128

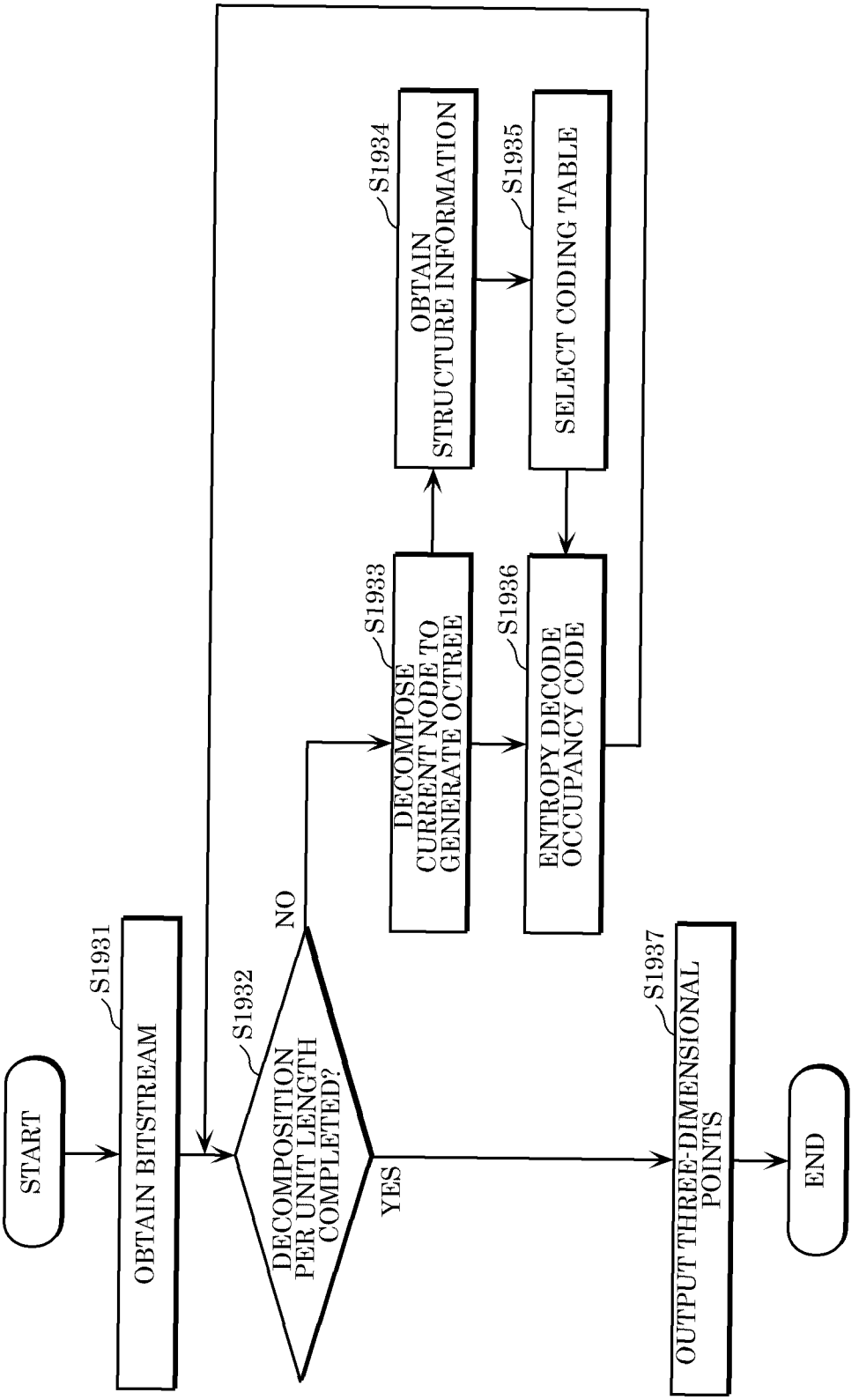

START
S1931
OBTAIN BITSTREAM
S1932
DECOMPOSITION PER UNIT LENGTH COMPLETED?
NO
YES
S1933
DECOMPOSE CURRENT NODE TO GENERATE OCTREE
S1934
OBTAIN STRUCTURE INFORMATION
S1935
SELECT CODING TABLE
S1936
ENTROPY DECODE OCCUPANCY CODE
S1937
OUTPUT THREE-DIMENSIONAL POINTS
END

FIG. 129

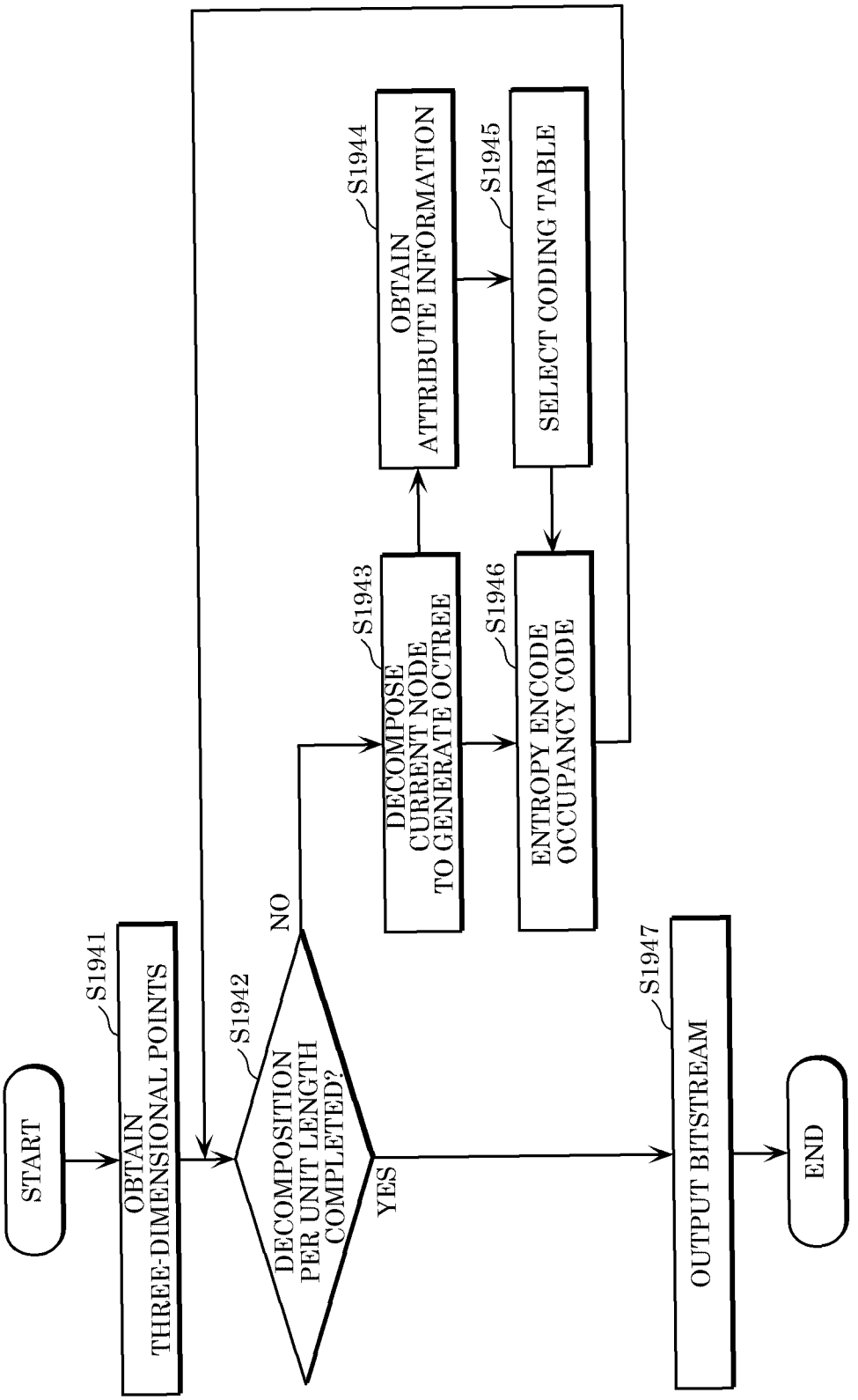

START
S1941
OBTAIN THREE-DIMENSIONAL POINTS
S1942
DECOMPOSITION PER UNIT LENGTH COMPLETED?
NO
YES
S1943
DECOMPOSE CURRENT NODE TO GENERATE OCTREE
S1944
OBTAIN ATTRIBUTE INFORMATION
S1945
SELECT CODING TABLE
S1946
ENTROPY ENCODE OCCUPANCY CODE
S1947
OUTPUT BITSTREAM
END

FIG. 130

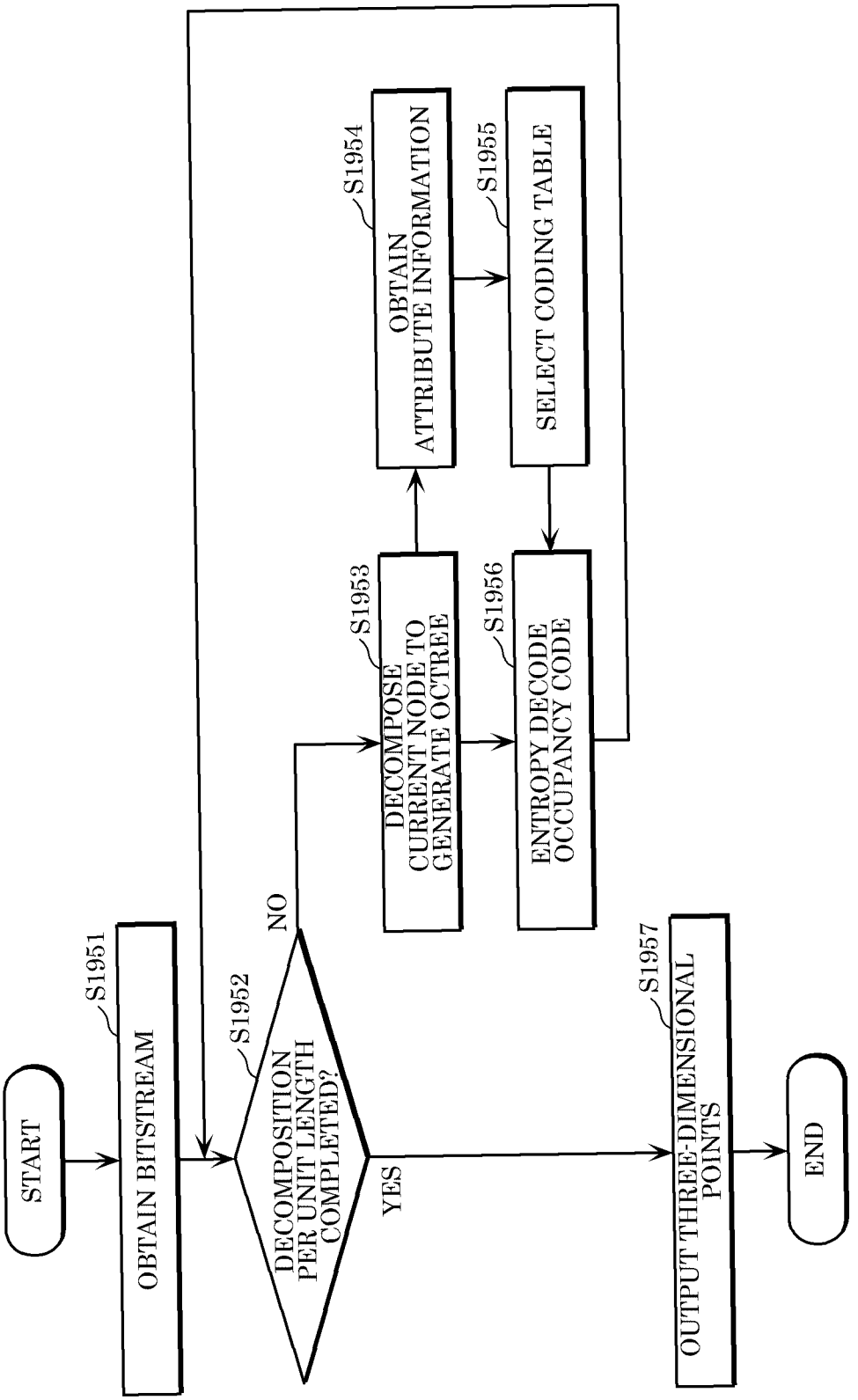

START
S1951
OBTAIN BITSTREAM
S1952
DECOMPOSITION PER UNIT LENGTH COMPLETED?
NO
YES
S1953
DECOMPOSE CURRENT NODE TO GENERATE OCTREE
S1954
OBTAIN ATTRIBUTE INFORMATION
S1955
SELECT CODING TABLE
S1956
ENTROPY DECODE OCCUPANCY CODE
S1957
OUTPUT THREE-DIMENSIONAL POINTS
END

FIG. 131

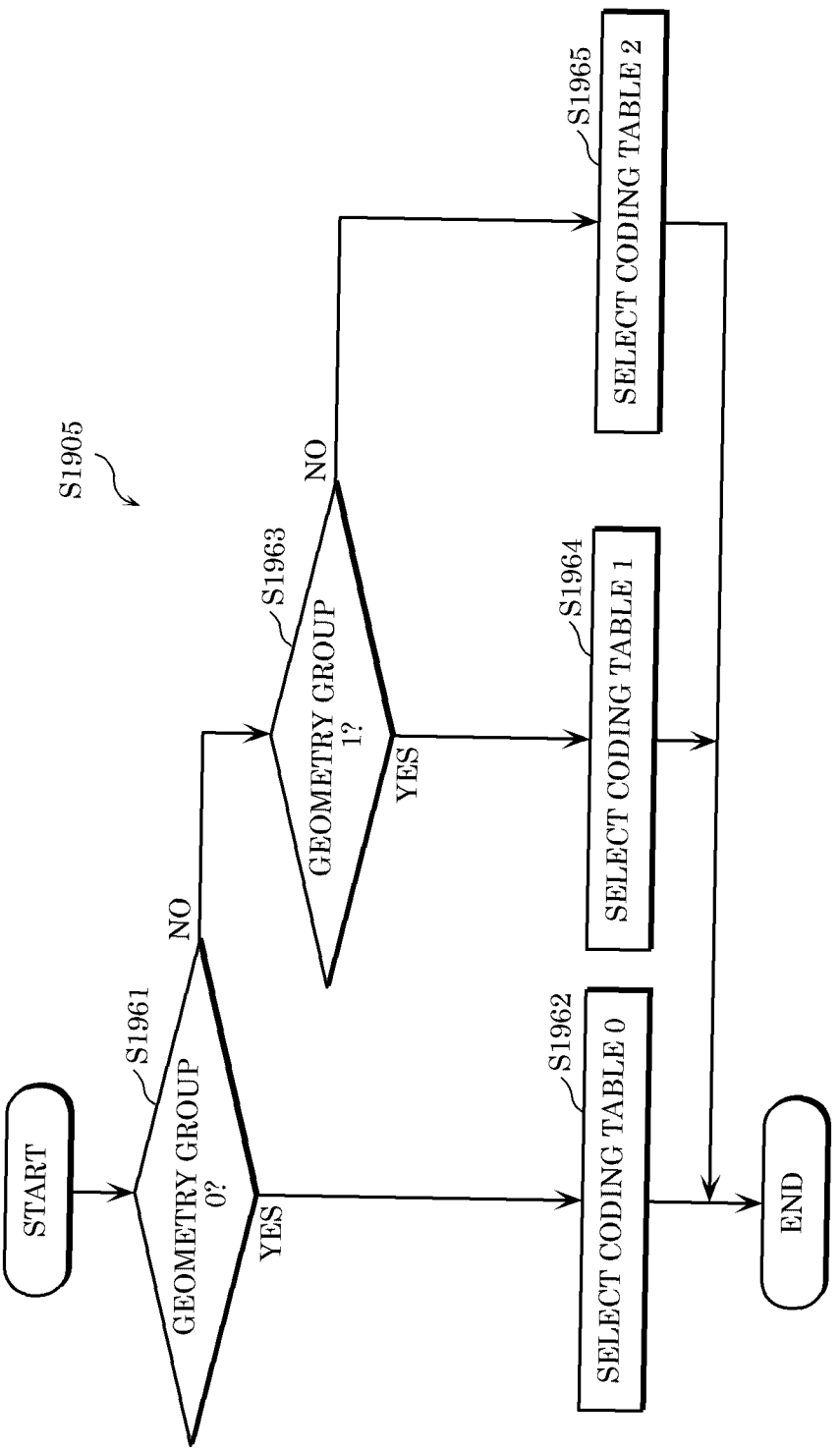

S1905
START
S1961
GEOMETRY GROUP 0?
YES
NO
S1962
SELECT CODING TABLE 0
S1963
GEOMETRY GROUP 1?
YES
NO
S1964
SELECT CODING TABLE 1
S1965
SELECT CODING TABLE 2
END

FIG. 132

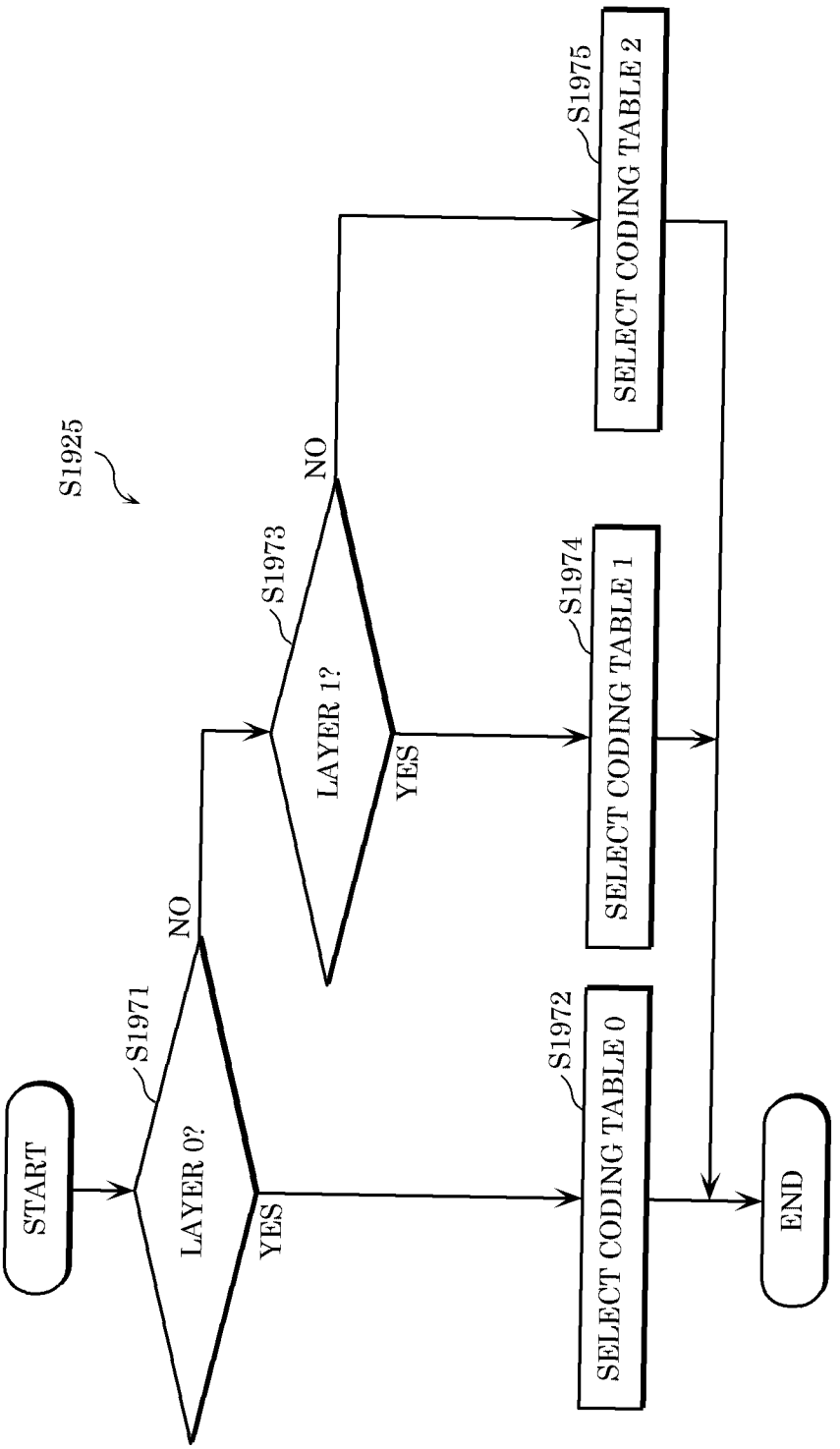

S1925
START
S1971
LAYER 0?
YES
NO
S1972
SELECT CODING TABLE 0
S1973
LAYER 1?
YES
NO
S1974
SELECT CODING TABLE 1
S1975
SELECT CODING TABLE 2
END

FIG. 133

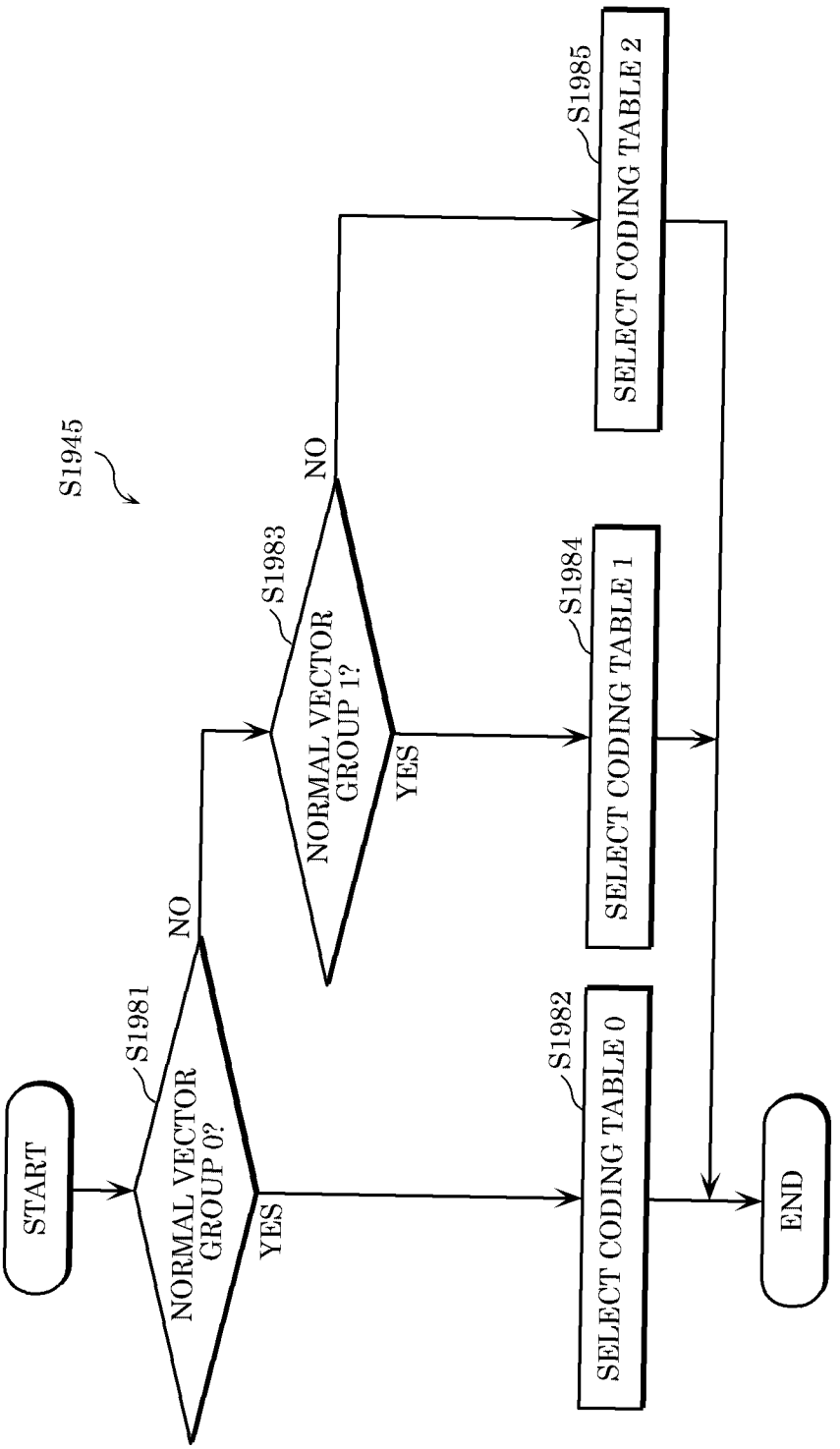

S1945
START
S1981
NORMAL VECTOR GROUP 0?
YES
NO
S1982
SELECT CODING TABLE 0
S1983
NORMAL VECTOR GROUP 1?
YES
NO
S1984
SELECT CODING TABLE 1
S1985
SELECT CODING TABLE 2
END

FIG. 134

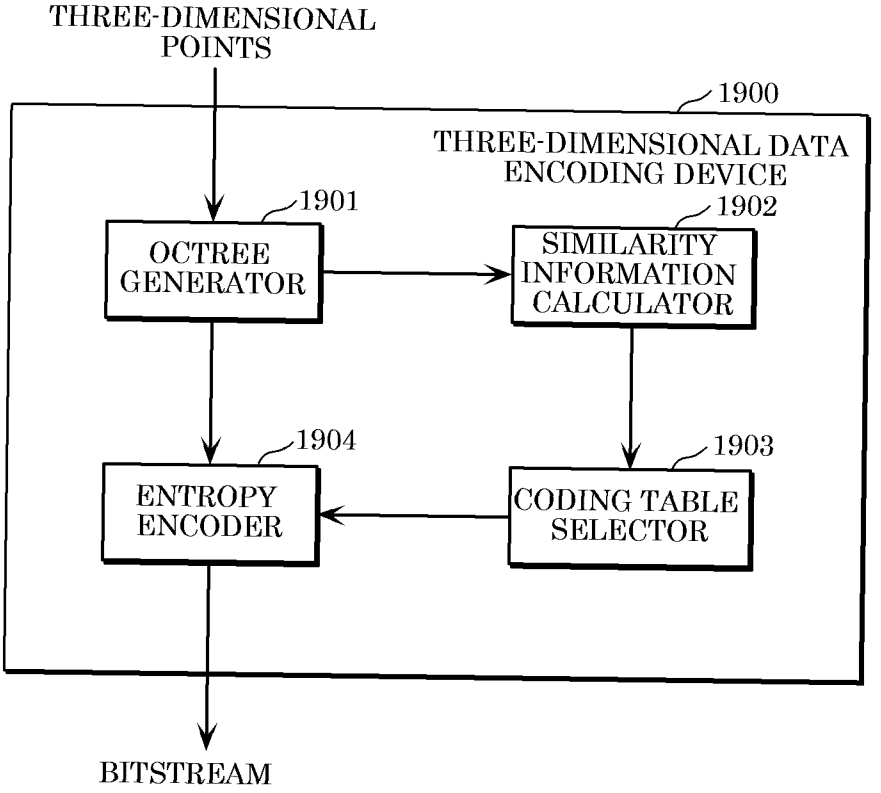


FIG. 135

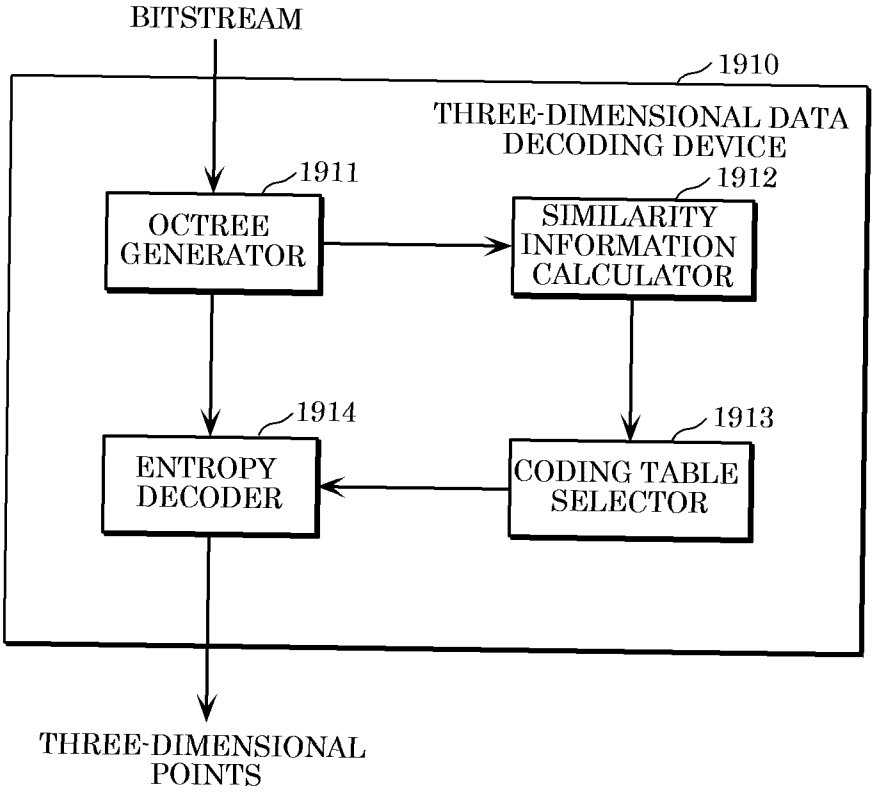


FIG. 136

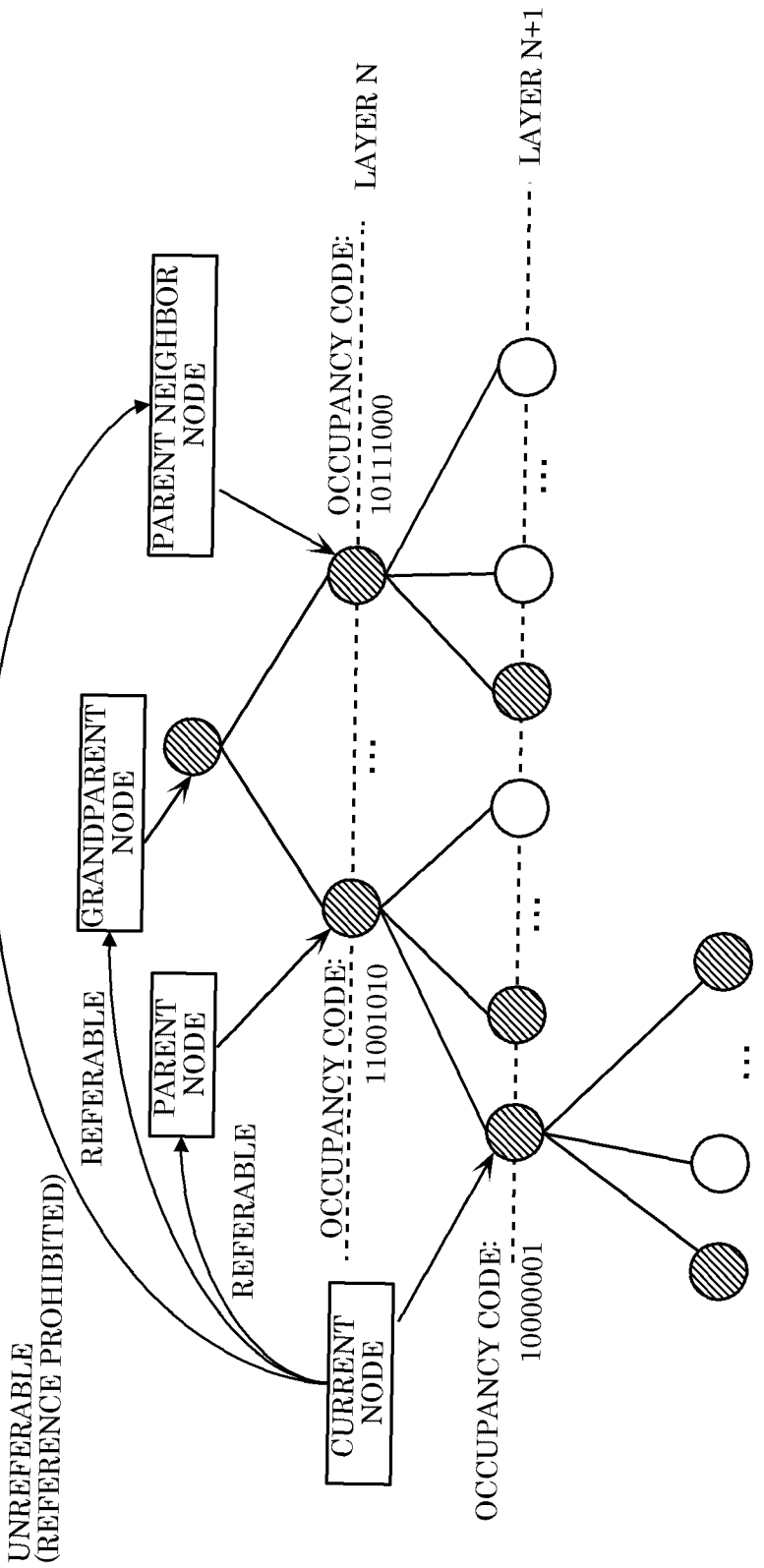

UNREFERABLE
(REFERENCE PROHIBITED)
REFERABLE
REFERABLE
CURRENT NODE
PARENT NODE
GRANDPARENT NODE
PARENT NEIGHBOR NODE
OCCUPANCY CODE: 10000001
OCCUPANCY CODE: 11001010
OCCUPANCY CODE: 10111000
LAYER N
LAYER N+1

FIG. 137

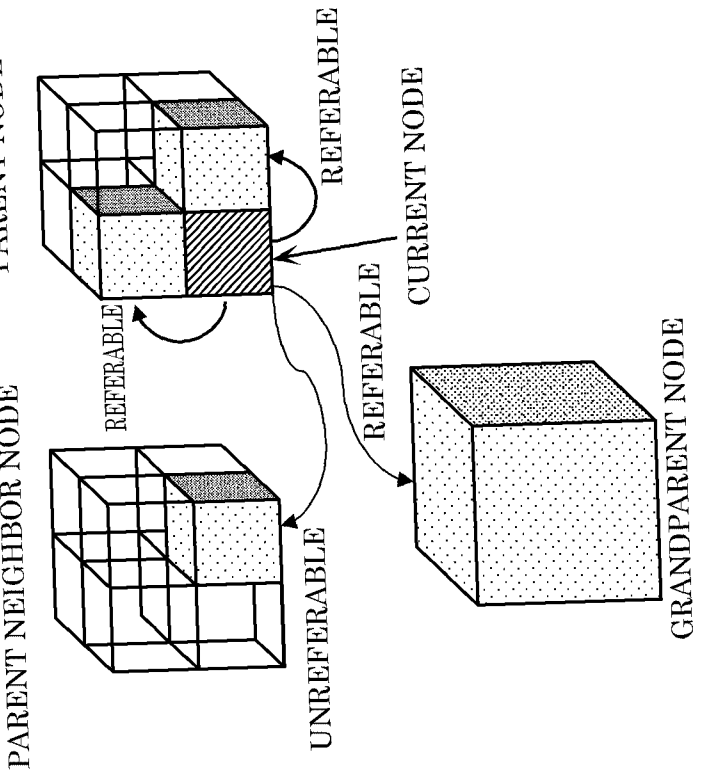

PARENT NODE
REFERABLE
CURRENT NODE
REFERABLE
REFERABLE
PARENT NEIGHBOR NODE
UNREFERABLE
GRANDPARENT NODE

FIG. 138

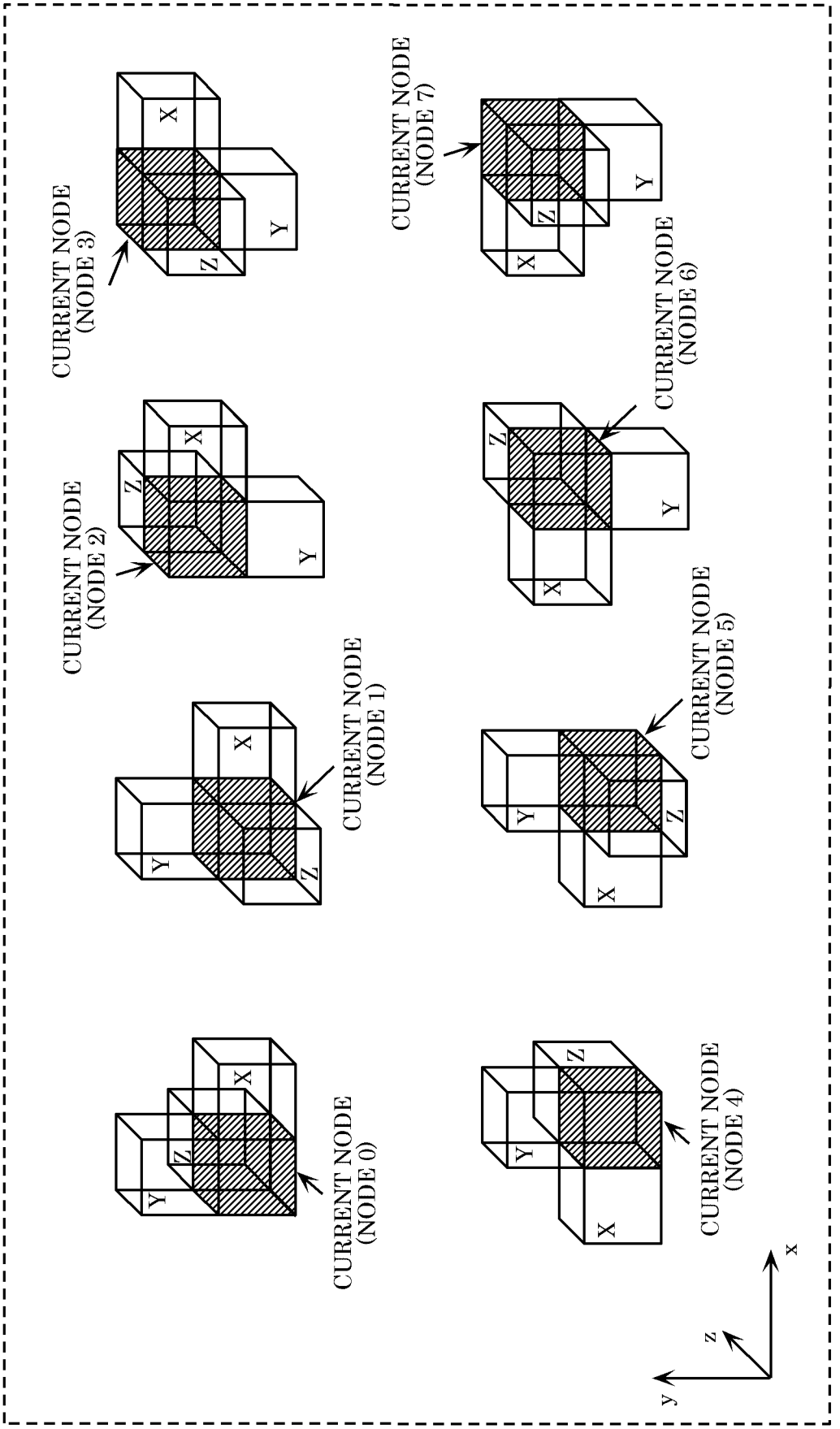

CURRENT NODE (NODE 0)
CURRENT NODE (NODE 1)
CURRENT NODE (NODE 2)
CURRENT NODE (NODE 3)
CURRENT NODE (NODE 4)
CURRENT NODE (NODE 5)
CURRENT NODE (NODE 6)
CURRENT NODE (NODE 7)
X
Y
Z
x
y
z

FIG. 139

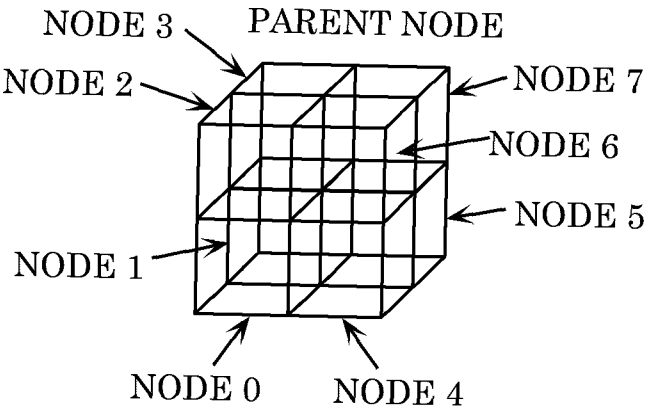


FIG. 140

EXAMPLE OF OCCUPANCY CODE OF PARENT NODE:

0 0 1 0 1 0 1 1

OCCUPANCY INFORMATION OF NODE 7

OCCUPANCY INFORMATION OF NODE 0

FIG. 141

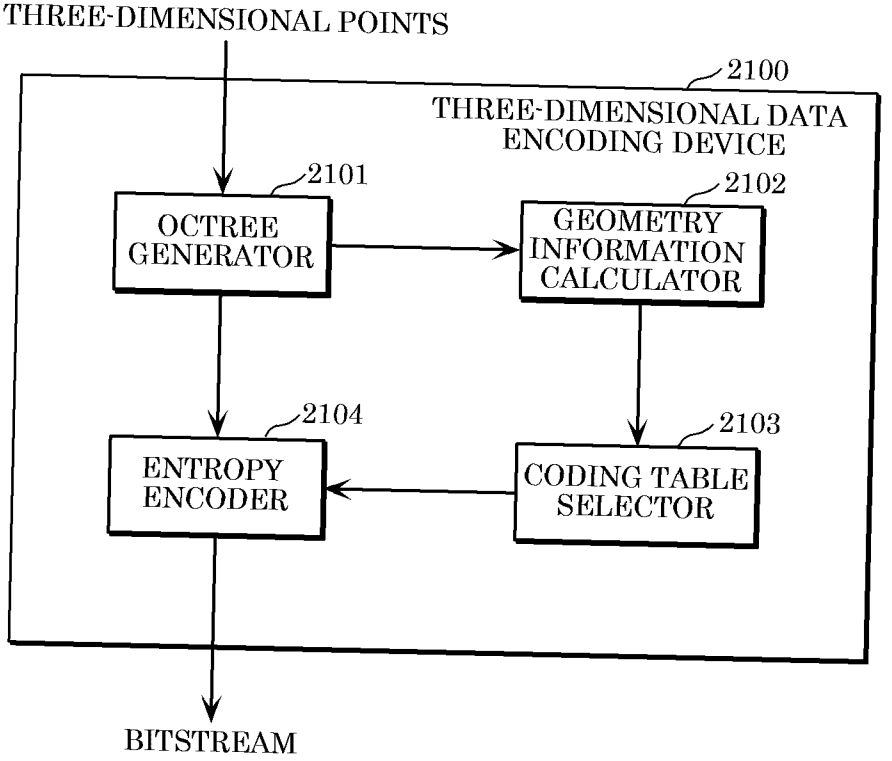


FIG. 142

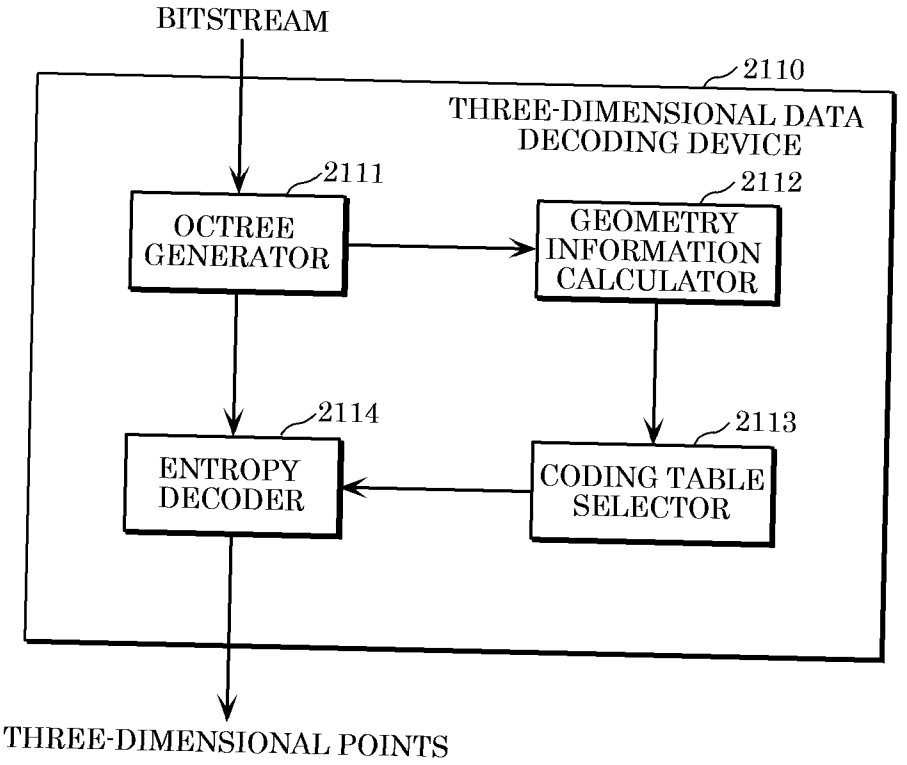

BITSTREAM
2110
THREE-DIMENSIONAL DATA DECODING DEVICE
2111
OCTREE GENERATOR
2112
GEOMETRY INFORMATION CALCULATOR
2114
ENTROPY DECODER
2113
CODING TABLE SELECTOR
THREE-DIMENSIONAL POINTS

FIG. 143

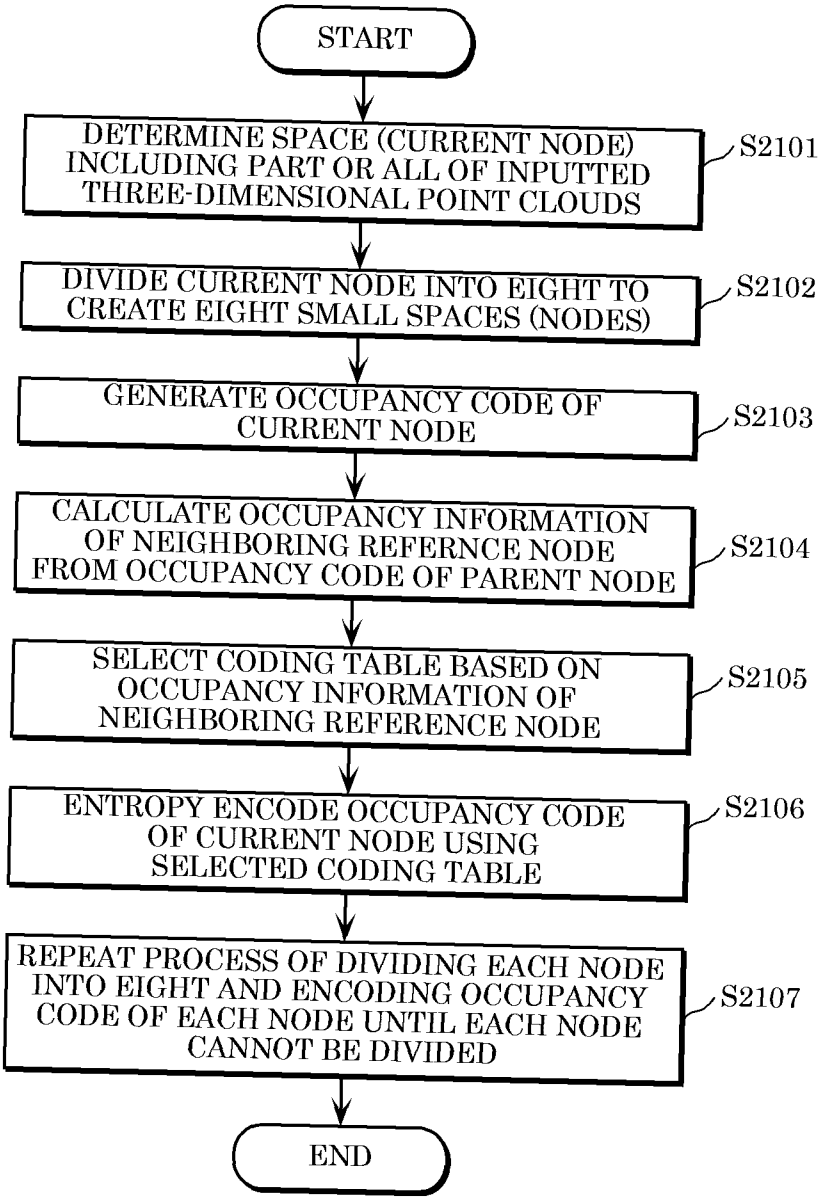

START
DETERMINE SPACE (CURRENT NODE) INCLUDING PART OR ALL OF INPUTTED THREE-DIMENSIONAL POINT CLOUDS
S2101
DIVIDE CURRENT NODE INTO EIGHT TO CREATE EIGHT SMALL SPACES (NODES)
S2102
GENERATE OCCUPANCY CODE OF CURRENT NODE
S2103
CALCULATE OCCUPANCY INFORMATION OF NEIGHBORING REFERNCE NODE FROM OCCUPANCY CODE OF PARENT NODE
S2104
SELECT CODING TABLE BASED ON OCCUPANCY INFORMATION OF NEIGHBORING REFERENCE NODE
S2105
ENTROPY ENCODE OCCUPANCY CODE OF CURRENT NODE USING SELECTED CODING TABLE
S2106
REPEAT PROCESS OF DIVIDING EACH NODE INTO EIGHT AND ENCODING OCCUPANCY CODE OF EACH NODE UNTIL EACH NODE CANNOT BE DIVIDED
S2107
END

FIG. 144

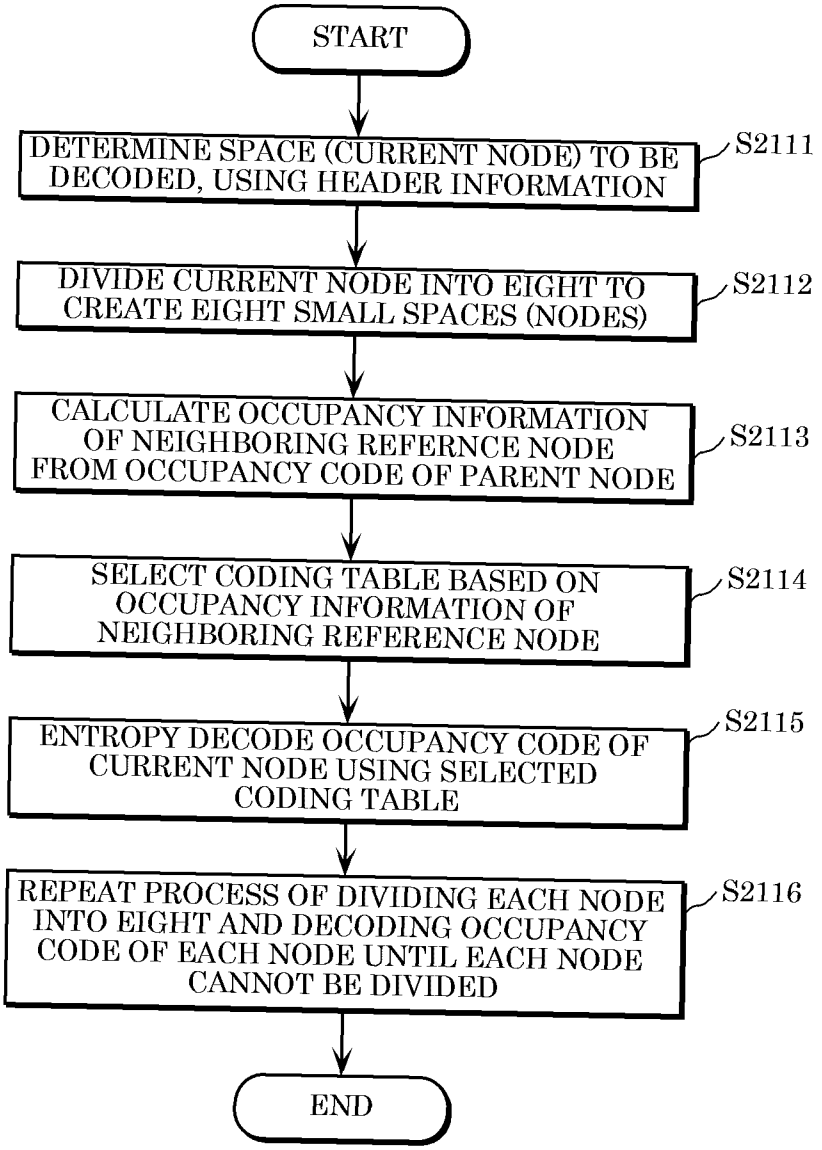

START
DETERMINE SPACE (CURRENT NODE) TO BE DECODED, USING HEADER INFORMATION
S2111
DIVIDE CURRENT NODE INTO EIGHT TO CREATE EIGHT SMALL SPACES (NODES)
S2112
CALCULATE OCCUPANCY INFORMATION OF NEIGHBORING REFERNCE NODE FROM OCCUPANCY CODE OF PARENT NODE
S2113
SELECT CODING TABLE BASED ON OCCUPANCY INFORMATION OF NEIGHBORING REFERENCE NODE
S2114
ENTROPY DECODE OCCUPANCY CODE OF CURRENT NODE USING SELECTED CODING TABLE
S2115
REPEAT PROCESS OF DIVIDING EACH NODE INTO EIGHT AND DECODING OCCUPANCY CODE OF EACH NODE UNTIL EACH NODE CANNOT BE DIVIDED
S2116
END

FIG. 145

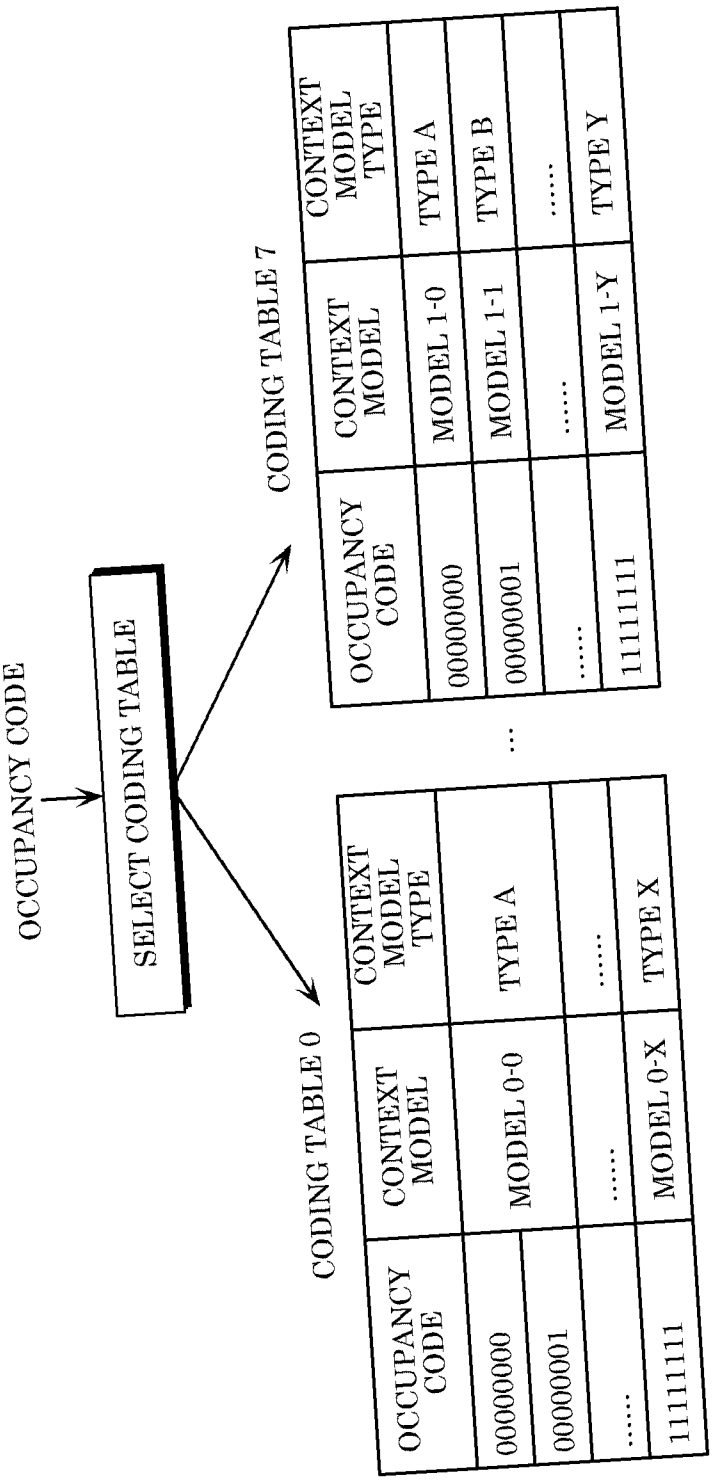

OCCUPANCY CODE
SELECT CODING TABLE
CODING TABLE 0
OCCUPANCY CODE
CONTEXT MODEL
CONTEXT MODEL TYPE
00000000
MODEL 0-0
TYPE A
00000001
.....
.....
11111111
MODEL 0-X
TYPE X
...
CODING TABLE 7
OCCUPANCY CODE
CONTEXT MODEL
CONTEXT MODEL TYPE
00000000
MODEL 1-0
TYPE A
00000001
MODEL 1-1
TYPE B
.....
.....
.....
11111111
MODEL 1-Y
TYPE Y

FIG. 146

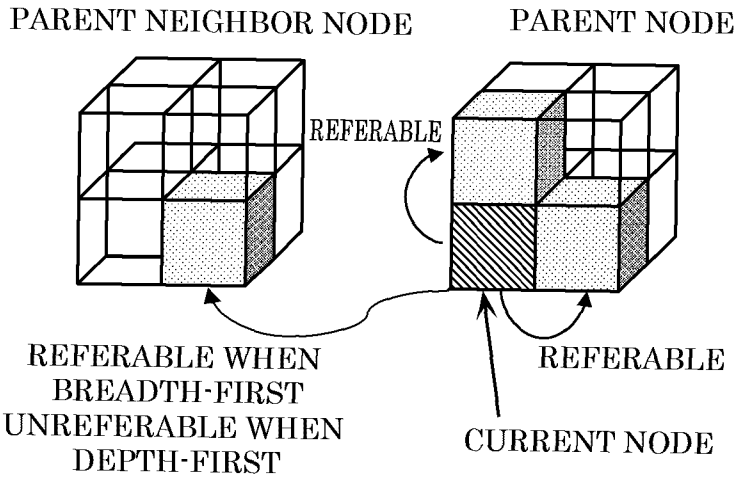


FIG. 147

```
pc_header () {
...
octree_scan_order
...
}
```

FIG. 148

```
pc_header () {
...
limit_refer_flag
...
}
```

FIG. 149

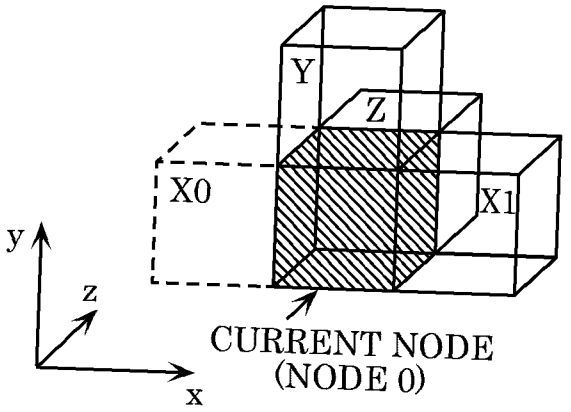


FIG. 150

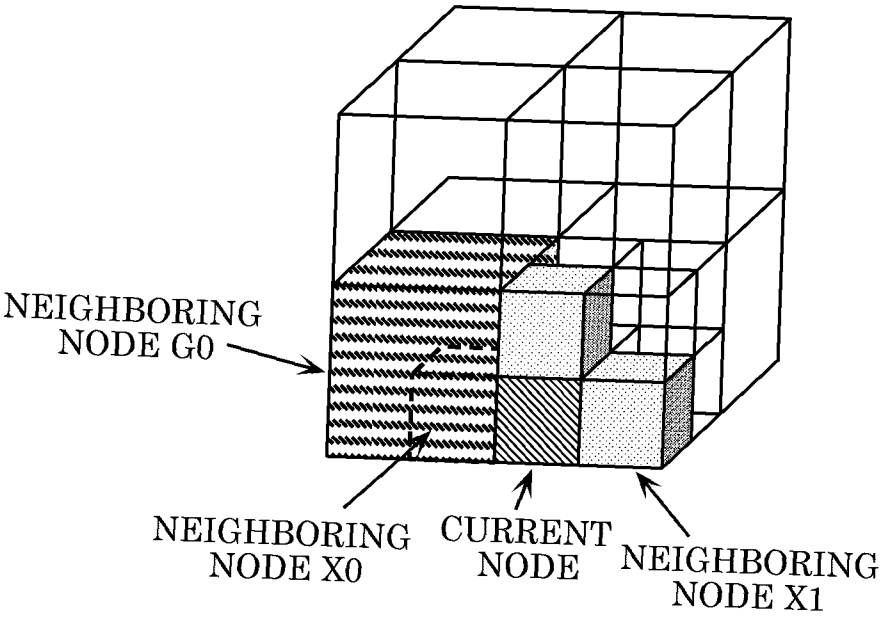


FIG. 151

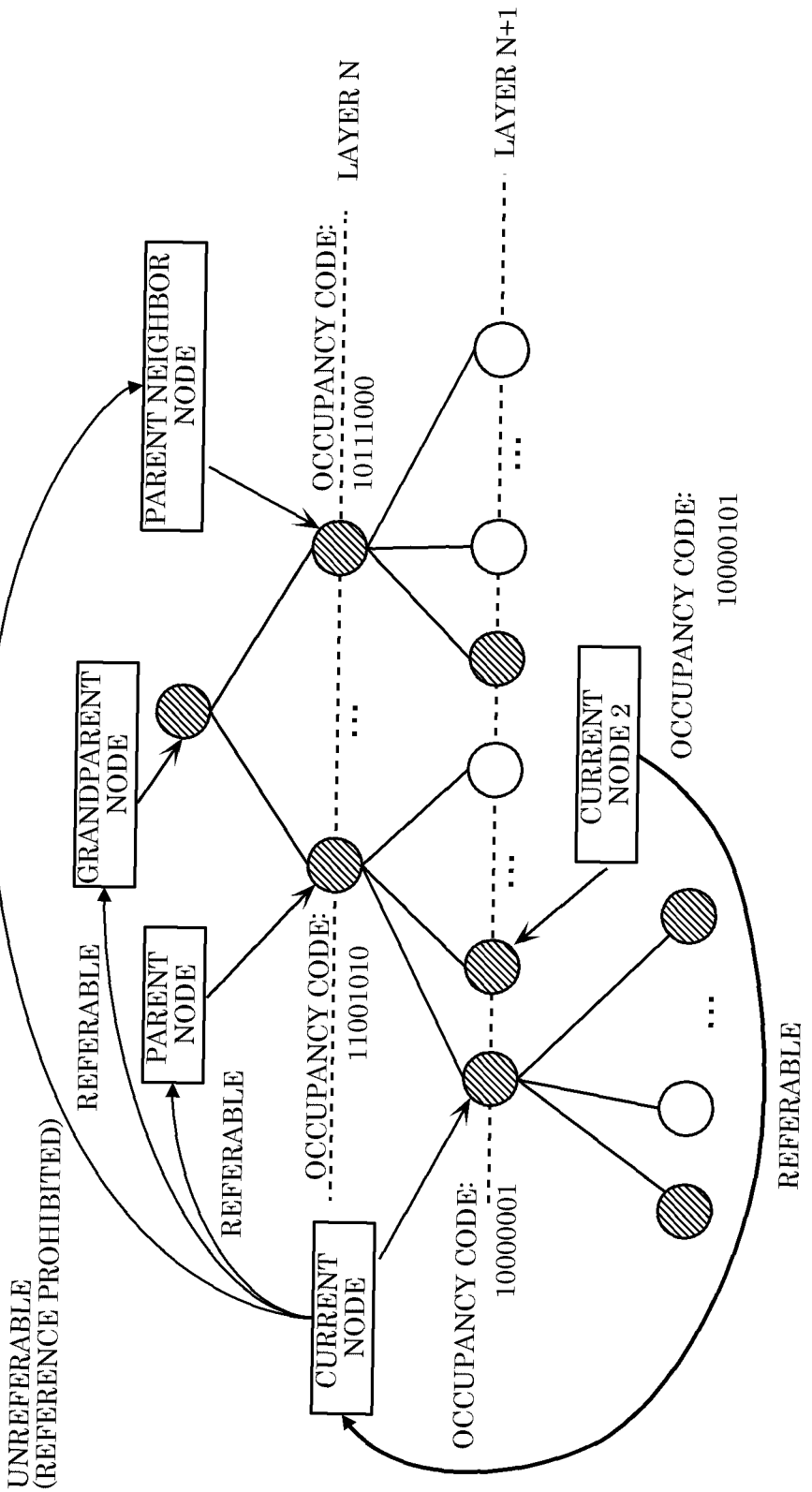

UNREFERABLE
(REFERENCE PROHIBITED)
REFERABLE
REFERABLE
REFERABLE
CURRENT NODE
PARENT NODE
GRANDPARENT NODE
PARENT NEIGHBOR NODE
CURRENT NODE 2
OCCUPANCY CODE: 10000001
OCCUPANCY CODE: 11001010
OCCUPANCY CODE: 10111000
OCCUPANCY CODE: 10000101
LAYER N
LAYER N+1

FIG. 152

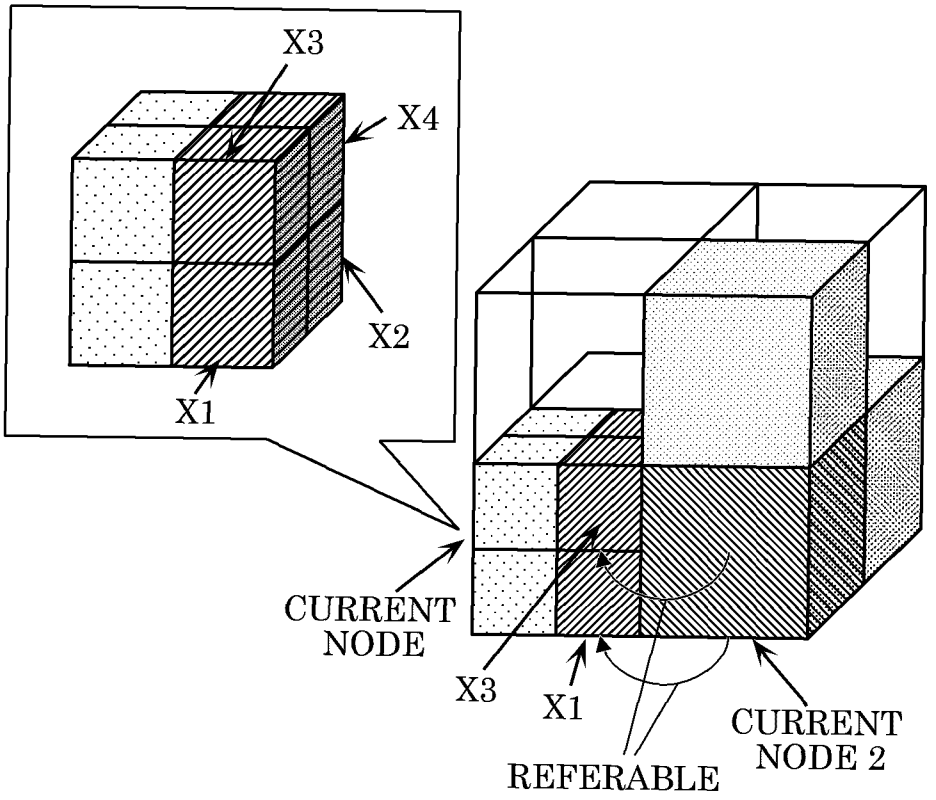


FIG. 153

```
pc_header () {
...
single_point_per_leaf
...
}
```

FIG. 154

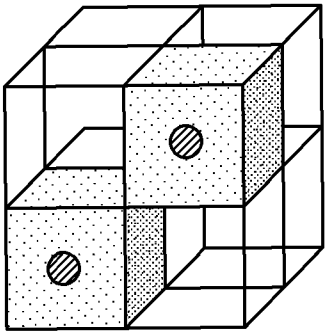


FIG. 155

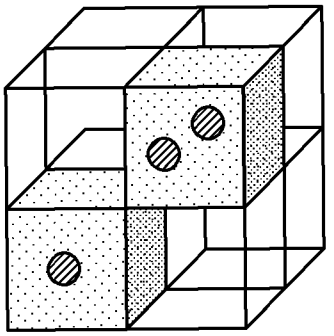


FIG. 156

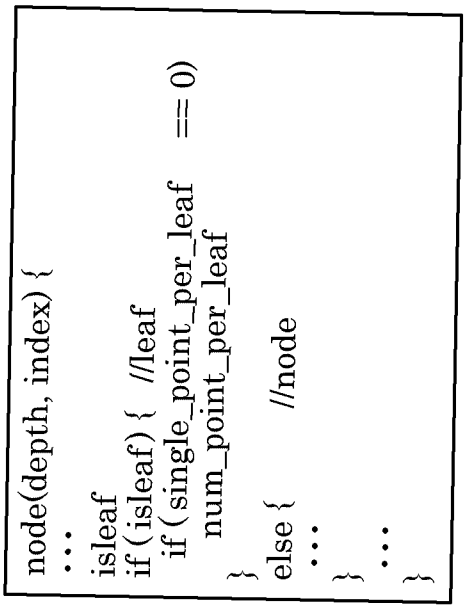

node(depth, index) {
...
isleaf
if (isleaf) { //leaf
if (single_point_per_leaf == 0)
num_point_per_leaf
}
else { //node
...
}
...
}

FIG. 157

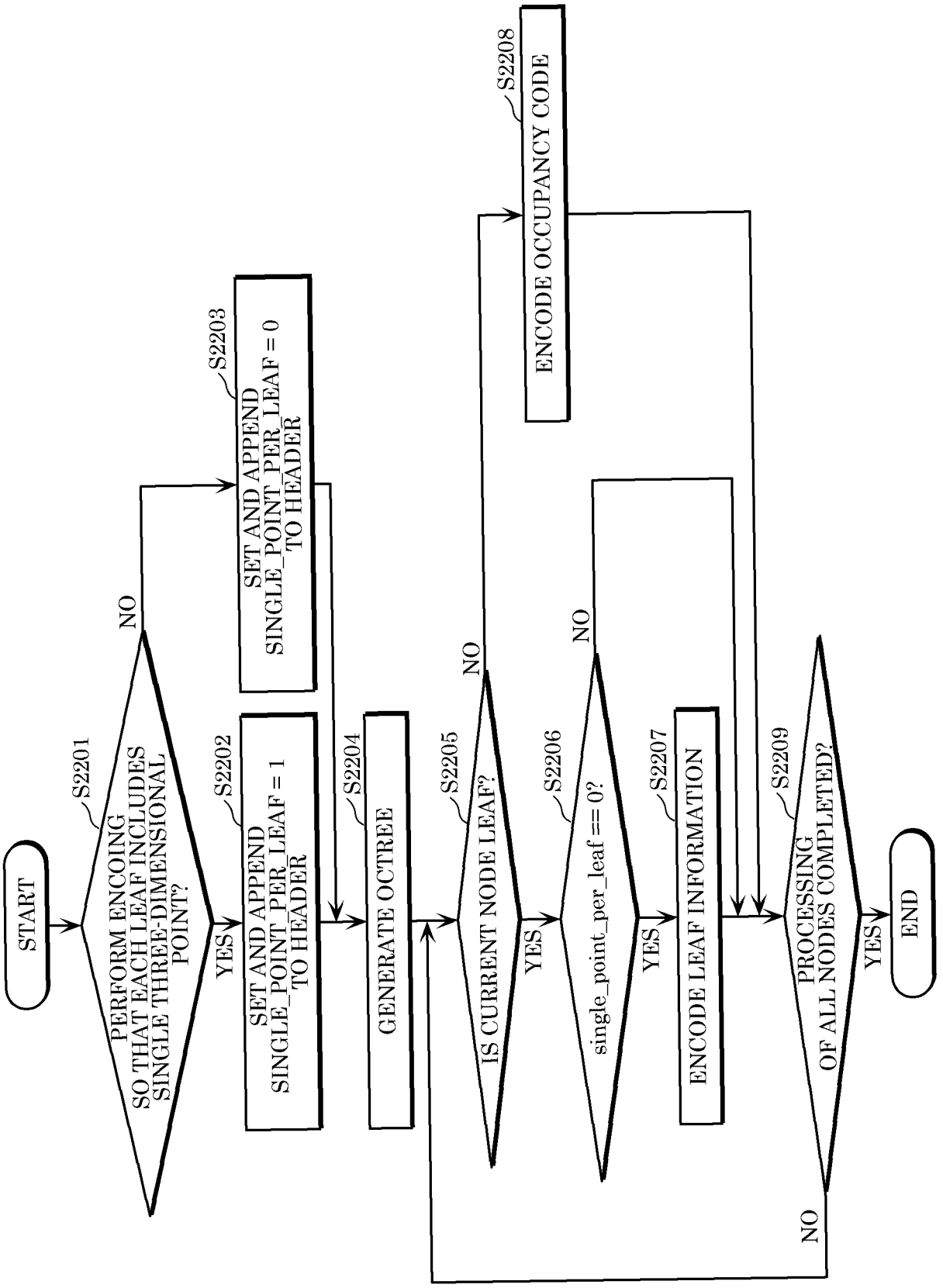

START
S2201
PERFORM ENCOING SO THAT EACH LEAF INCLUDES SINGLE THREE-DIMENSIONAL POINT?
NO
YES
S2202
SET AND APPEND SINGLE_POINT_PER_LEAF = 1 TO HEADER
S2203
SET AND APPEND SINGLE_POINT_PER_LEAF = 0 TO HEADER
S2204
GENERATE OCTREE
S2205
IS CURRENT NODE LEAF?
NO
YES
S2206
single_point_per_leaf == 0?
NO
YES
S2207
ENCODE LEAF INFORMATION
S2208
ENCODE OCCUPANCY CODE
S2209
PROCESSING OF ALL NODES COMPLETED?
NO
YES
END

FIG. 158

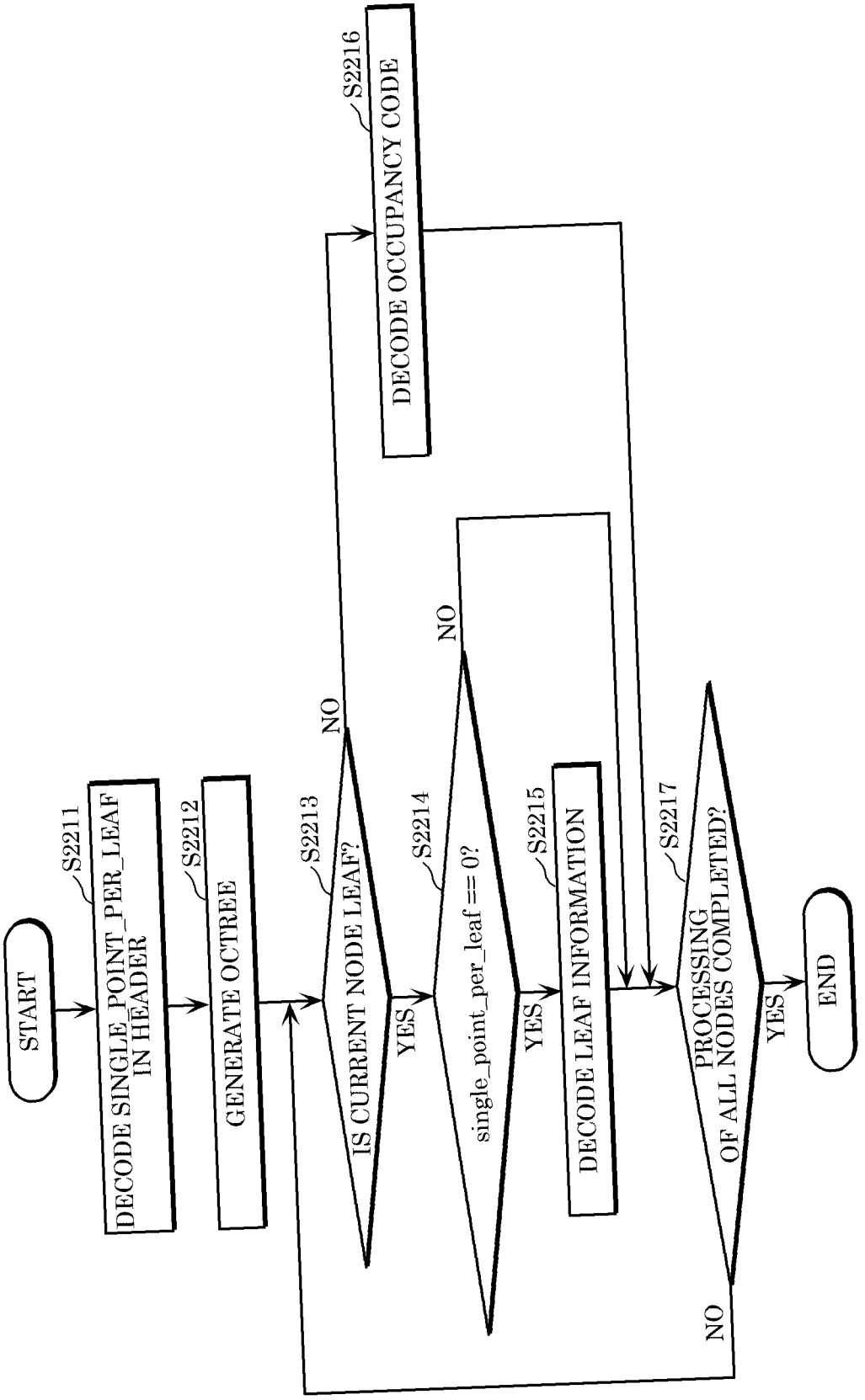

START
S2211
DECODE SINGLE_POINT_PER_LEAF IN HEADER
S2212
GENERATE OCTREE
S2213
IS CURRENT NODE LEAF?
NO
YES
S2214
single_point_per_leaf == 0?
NO
YES
S2215
DECODE LEAF INFORMATION
S2216
DECODE OCCUPANCY CODE
S2217
PROCESSING OF ALL NODES COMPLETED?
NO
YES
END

FIG. 159

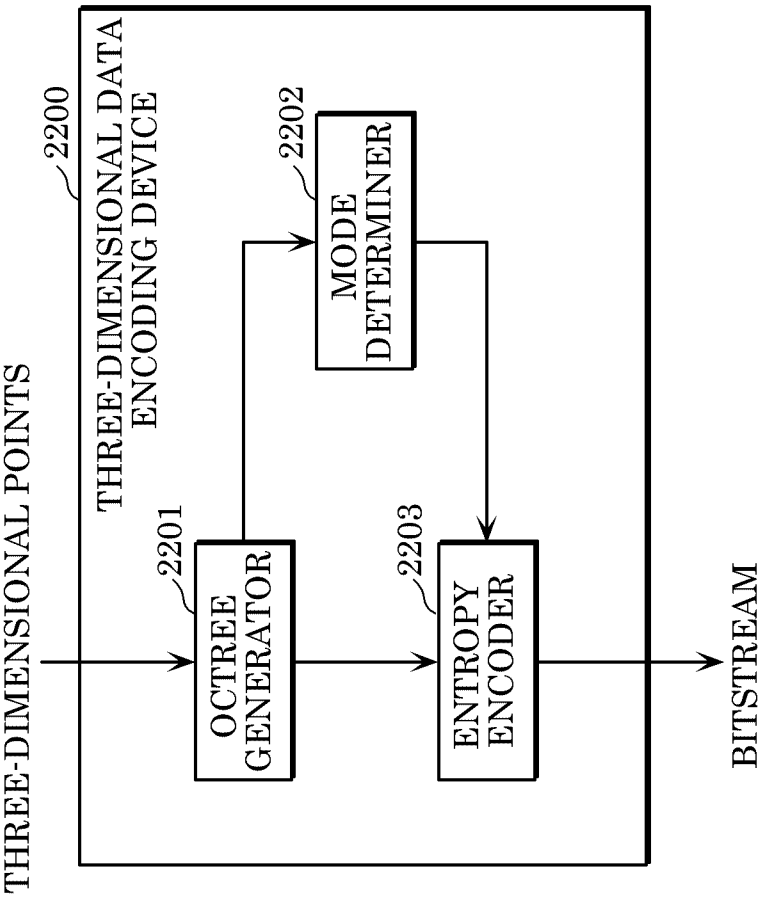

THREE-DIMENSIONAL POINTS
2200
THREE-DIMENSIONAL DATA ENCODING DEVICE
2201
OCTREE GENERATOR
2202
MODE DETERMINER
2203
ENTROPY ENCODER
BITSTREAM

FIG. 160

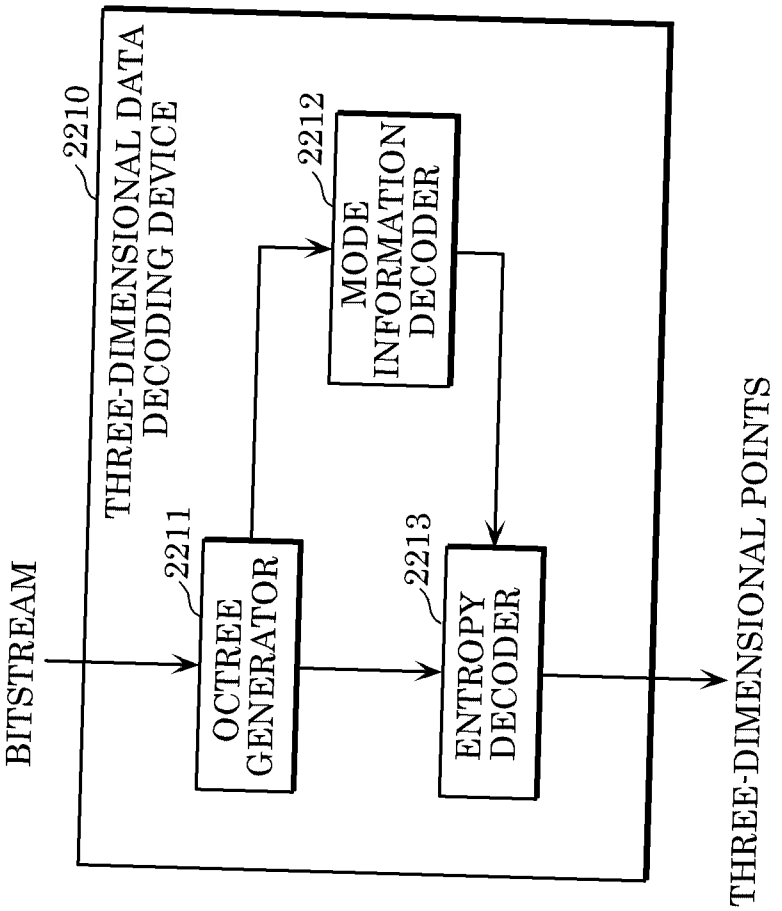

BITSTREAM
2210
THREE-DIMENSIONAL DATA DECODING DEVICE
2211
OCTREE GENERATOR
2212
MODE INFORMATION DECODER
2213
ENTROPY DECODER
THREE-DIMENSIONAL POINTS

FIG. 161

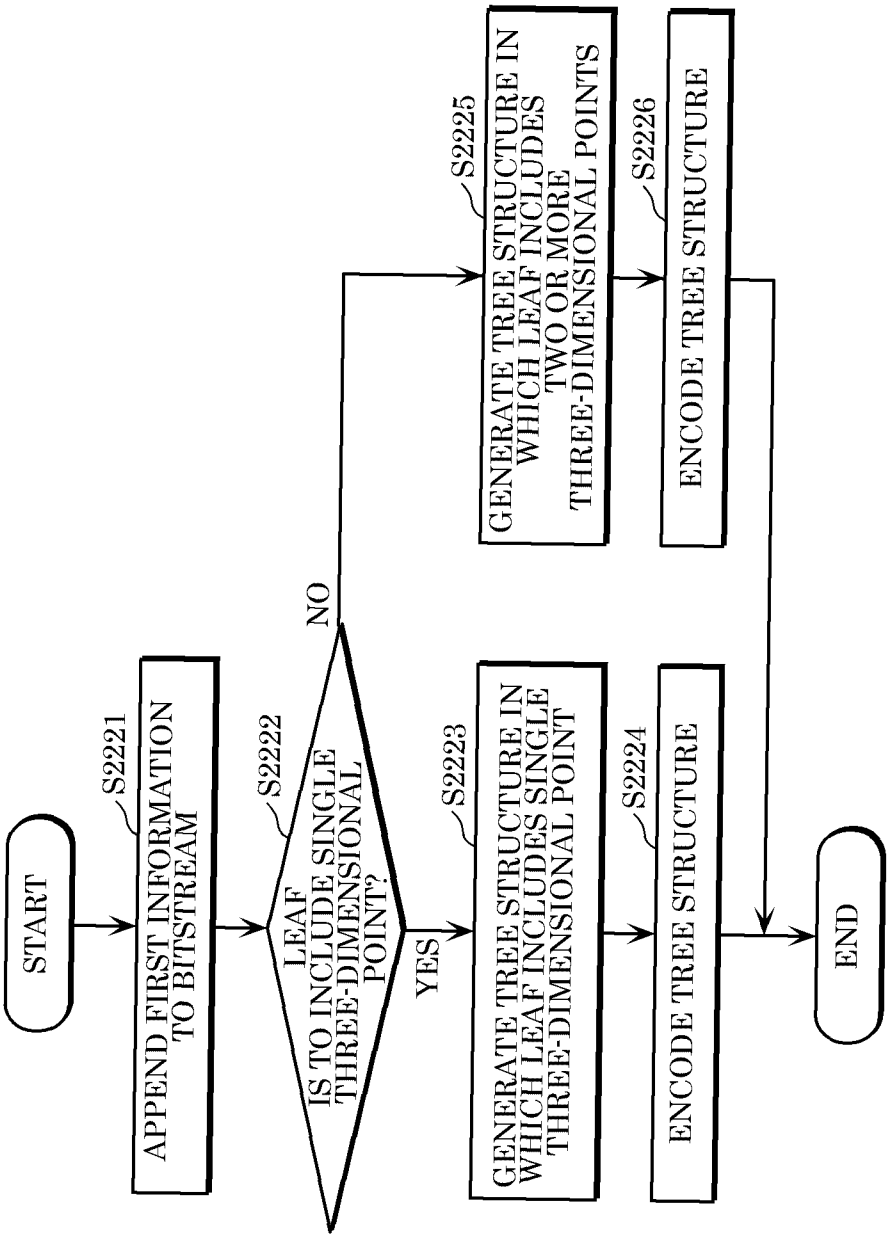
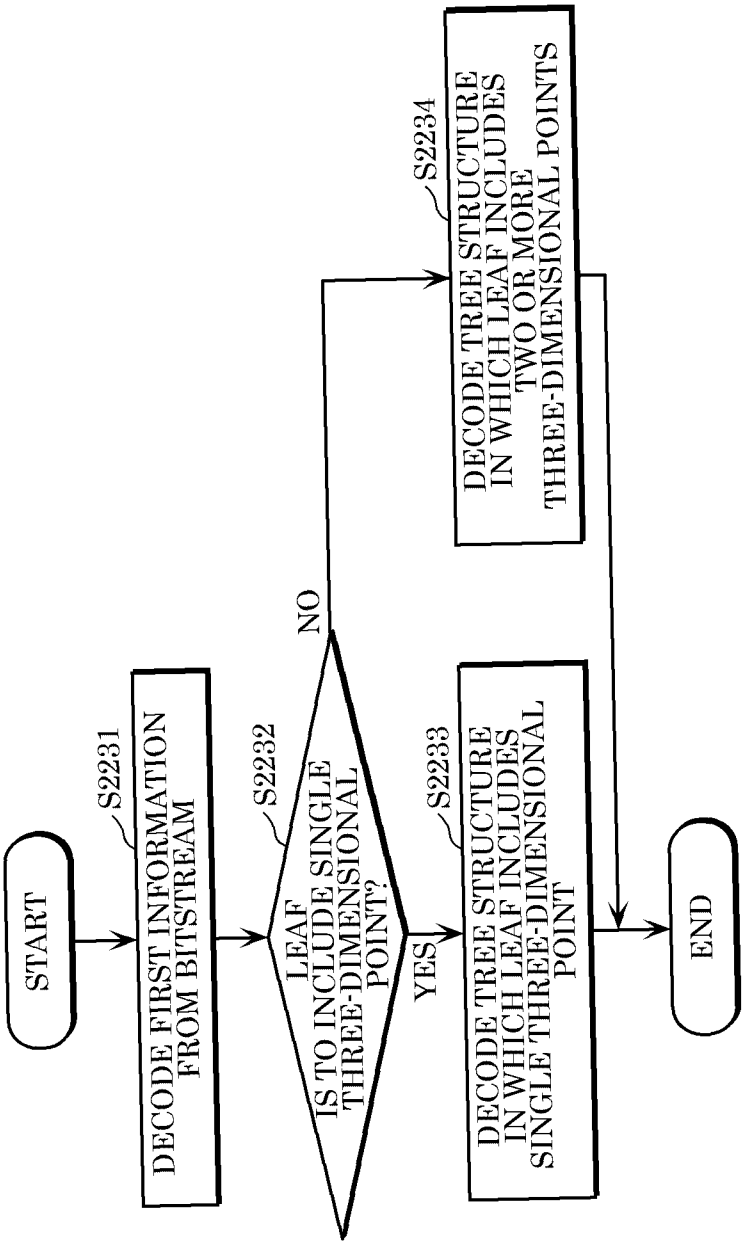
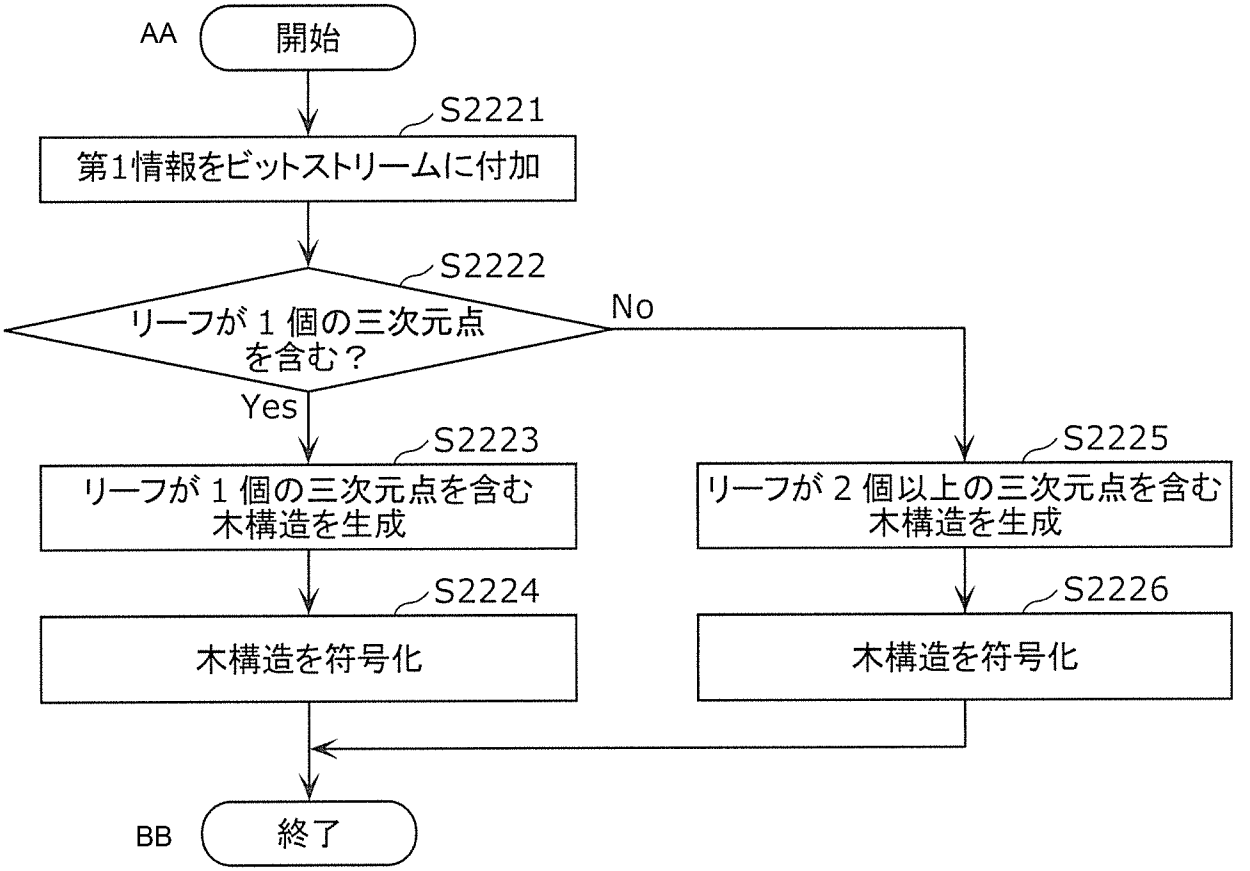

START
S2221
APPEND FIRST INFORMATION TO BITSTREAM
S2222
IS TO INCLUDE SINGLE THREE-DIMENSIONAL POINT?
YES
NO
S2223
GENERATE TREE STRUCTURE IN WHICH LEAF INCLUDES SINGLE THREE-DIMENSIONAL POINT
S2224
ENCODE TREE STRUCTURE
S2225
GENERATE TREE STRUCTURE IN WHICH LEAF INCLUDES TWO OR MORE THREE-DIMENSIONAL POINTS
S2226
ENCODE TREE STRUCTURE
END

FIG. 162


START
S2231
DECODE FIRST INFORMATION FROM BITSTREAM
S2232
IS TO INCLUDE SINGLE THREE-DIMENSIONAL POINT?
YES
NO
S2233
DECODE TREE STRUCTURE IN WHICH LEAF INCLUDES SINGLE THREE-DIMENSIONAL POINT
S2234
DECODE TREE STRUCTURE IN WHICH LEAF INCLUDES TWO OR MORE THREE-DIMENSIONAL POINTS
END



S2221 Add first information to bitstream
S2222 Does leaf include one three-dimensional point?
S2223 Generate tree structure in which leaf includes one three-dimensional point
S2224, S2226 Encode tree structure
S2225 Generate tree structure in which leaf includes two or more three-dimensional points
AA Start
BB End