

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **3 001 192**

51 Int. Cl.:

G06T 9/00 (2006.01)

H03M 7/40 (2006.01)

G06T 9/40 (2006.01)

H04N 19/96 (2014.01)

H04N 19/91 (2014.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **02.10.2018** **E 23169774 (9)**

97 Fecha y número de publicación de la concesión europea: **04.12.2024** **EP 4231241**

54 Título: **Métodos y dispositivos para la codificación entrópica binaria de nubes de puntos**

30 Prioridad:

18.01.2018 EP 18305037

45 Fecha de publicación y mención en BOPI de la traducción de la patente:

04.03.2025

73 Titular/es:

MALIKIE INNOVATIONS LIMITED (100.00%)
The Glasshouses GH2, 92 Georges Street Lower
Dun Laoghaire, Dublin A96 VR66, IE

72 Inventor/es:

LASSERRE, SÉBASTIEN

74 Agente/Representante:

ELZABURU, S.L.P

ES 3 001 192 T3

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Métodos y dispositivos para la codificación entrópica binaria de nubes de puntos

Campo

- 5 La presente solicitud se refiere en general a la compresión de nube de puntos y, en particular, a métodos y dispositivos para la codificación entrópica binaria de nubes de puntos.

Antecedentes

- 10 En comunicaciones y redes informáticas se usa compresión de datos para almacenar, transmitir y reproducir información de manera eficiente. Existe un interés creciente en representaciones de objetos o espacios tridimensionales, que pueden implicar grandes conjuntos de datos y para los que una compresión eficiente y eficaz sería muy útil y valorada. En algunos casos, los objetos o espacios tridimensionales pueden representarse usando una nube de puntos, que es un conjunto de puntos que tienen cada uno una localización de tres coordenadas (X, Y, Z) y, en algunos casos, otros atributos como datos de color (p. ej. luminancia y crominancia), transparencia, reflectancia, vector normal, etc. Las nubes de puntos pueden ser estáticas (un objeto estacionario o una instantánea de un entorno/objeto en un único punto en el tiempo) o dinámicas (una secuencia ordenada en el tiempo de nubes de puntos).
- 15 Aplicaciones de ejemplo para nubes de puntos incluyen aplicaciones de topografía y mapeo. Aplicaciones de vehículo autónomo y otras de visión de máquina pueden basarse en datos de sensor de nube de puntos en forma de escaneos 3D de un entorno, tal como de un escáner LiDAR. Las simulaciones de realidad virtual pueden basarse en nubes de puntos.
- 20 Se apreciará que las nubes de puntos pueden implicar grandes cantidades de datos y comprimir (codificar y decodificar) esos datos rápidamente y con precisión es de interés significativo. Por consiguiente, sería ventajoso proporcionar métodos y dispositivos que compriman datos de manera más eficiente y/o efectiva para nubes de puntos. Además, sería ventajoso encontrar métodos y dispositivos para codificar nubes de puntos que puedan implementarse usando codificación entrópica binaria adaptativa al contexto sin requerir la gestión de un número excesivo de contextos.

Breve descripción de los dibujos

- 25 A continuación, se hará referencia, a modo de ejemplo, a los dibujos adjuntos que muestran realizaciones de ejemplo de la presente solicitud, y en donde:
- la Figura 1 muestra un diagrama de bloques simplificado de un codificador de nube de puntos de ejemplo;
 - la Figura 2 muestra un diagrama de bloques simplificado de un decodificador de nube de puntos de ejemplo;
 - la Figura 3 muestra un subvolumen parcial de ejemplo y una estructura de árbol asociada para codificación;
 - 30 la Figura 4 ilustra la partición y codificación recursivas de un árbol octal;
 - la Figura 5 muestra un patrón de exploración de ejemplo dentro de un cubo de ejemplo de un árbol octal;
 - la Figura 6 muestra un patrón de ocupación de ejemplo dentro de un cubo de ejemplo;
 - la Figura 7 muestra, en forma de diagrama de flujo, un método de ejemplo para codificar una nube de puntos;
 - la Figura 8 ilustra una parte de un árbol octal de ejemplo;
 - 35 la Figura 9 muestra un ejemplo de subvolúmenes vecinos;
 - la Figura 10 muestra una configuración de vecino de ejemplo que muestra la ocupación entre nodos vecinos;
 - la Figura 11 muestra esquemáticamente una realización ilustrativa de un proceso de codificación entrópica de nube de puntos usando contexto dependiente de patrón padre;
 - la Figura 12 muestra una realización ilustrativa de un proceso de codificación entrópica en la nube de puntos usando contexto dependiente de la configuración de vecino;
 - 40 la Figura 13 muestra, en forma de diagrama de flujo, un método de ejemplo para decodificar un flujo de bits de datos comprimidos en la nube de puntos;
 - la Figura 14 muestra un diagrama de bloques simplificado de ejemplo de un codificador;
 - la Figura 15 muestra un diagrama de bloques simplificado de ejemplo de un decodificador;
 - 45 la Figura 16 muestra un sistema de coordenadas cartesianas de ejemplo y rotaciones y/o reflexiones de ejemplo

alrededor de los ejes;

la Figura 17 muestra clases de invariancia de la configuración de vecino bajo una o varias iteraciones de la rotación alrededor del eje Z;

la Figura 18 muestra clases de invariancia de la configuración de vecino para reflexión vertical;

5 la Figura 19 muestra clases de invariancia tanto para rotación como para reflexión;

la Figura 20 muestra clases de invariancia para menos de tres rotaciones y la reflexión;

la Figura 21 ilustra la equivalencia entre la codificación no binaria y la codificación binaria en cascada para un patrón de ocupación;

10 la Figura 22 muestra en forma de diagrama de flujo, un método de ejemplo para codificar patrones de ocupación en un codificador en la nube de puntos basado en árbol usando codificación binaria;

la Figura 23 muestra un diagrama de bloques simplificado de parte de un codificador de ejemplo;

la Figura 24 muestra esquemáticamente una operación de reducción de contexto de ejemplo basada en la cribado de vecinos;

la Figura 25 muestra otra operación de reducción de contexto de ejemplo basada en la cribado de vecinos;

15 la Figura 26 muestra un ejemplo, en forma de diagrama de flujo, de un método de patrones de ocupación de codificación binaria usando reducción de contexto combinada;

la Figura 27 muestra un ejemplo de subvolumenes vecinos entre los que algunos ya están codificados;

la Figura 28 muestra un ejemplo de subvolumenes vecinos y subvolumenes ya codificados de los subvolumenes vecinos;

20 la Figura 29 muestra un ejemplo de ocupación de subvolumenes en un subvolumen vecino;

la Figura 30 muestra, en forma de diagrama de flujo, un método de codificación de un patrón de ocupación de un nodo actual basándose, al menos en parte, en datos de ocupación para nodos hijos de al menos uno de una pluralidad de nodos vecinos;

25 la Figura 31 muestra, en forma de diagrama de flujo, un método de decodificación de un patrón de ocupación de un nodo actual basándose, al menos en parte, en datos de ocupación para nodos hijos de al menos uno de una pluralidad de nodos vecinos;

la Figura 32 muestra, en forma de diagrama de flujo, un método para decidir la configuración de vecino dependiendo de los subvolumenes de subvolumenes vecinos;

la Figura 33 muestra otro ejemplo de ocupación de subvolumenes en un subvolumen vecino; y

30 la Figura 34 muestra otro ejemplo más de ocupación de subvolumenes en un subvolumen vecino.

En diferentes figuras se pueden haber usado números de referencia similares para indicar componentes similares.

Descripción de realizaciones de ejemplo

35 La presente solicitud describe métodos de codificación y decodificación de nubes de puntos, y codificadores y decodificadores para codificar y decodificar nubes de puntos. Una secuencia de bits que señala un patrón de ocupación para subvolumenes de un volumen puede codificarse usando codificación entrópica (por ejemplo, codificación entrópica binaria). Las probabilidades asociadas con los respectivos codificadores entrópicos para su uso en la codificación entrópica, el patrón de ocupación puede seleccionarse en función de datos de ocupación para subvolumenes vecinos de un subvolumen actual y además en función de datos de ocupación para subvolumenes hijos de al menos uno de los subvolumenes vecinos.

40 En ejemplos que son útiles para entender la aplicación, los contextos pueden basarse en la configuración de vecino y una secuencia parcial de bits previamente codificados de la secuencia de bits. Se puede hacer una determinación en cuanto a si aplicar una operación de reducción de contexto y, si es así, la operación reduce el número de contextos disponibles. Las operaciones de reducción de contexto de ejemplo incluyen reducir configuraciones de vecino basadas en apantallamiento por subvolumenes asociados con bits previamente codificados, manejo especial para configuraciones de vecino vacías y consolidación de contexto basada en estadísticas. La reducción puede aplicarse antes de la codificación y puede realizarse una determinación durante la codificación en cuanto a si se cumplen las circunstancias para usar un conjunto de contexto reducido.

La invención se define por la materia de asunto de las reivindicaciones independientes adjuntas 1, 2 y 9-12.

En un aspecto, la presente solicitud proporciona un método para codificar una nube de puntos para generar un flujo de bits de datos de nube de puntos comprimidos, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolumenes y que contiene los puntos de la nube de puntos. El método incluye, para un nodo actual asociado con un subvolumen partido en subvolumenes adicionales, correspondiendo cada subvolumen adicional a un nodo hijo del nodo actual, determinar un patrón de ocupación para el nodo actual en función de los estados de ocupación de los nodos hijos. El método incluye además seleccionar una o más probabilidades asociadas con los respectivos codificadores entrópicos para codificar por entropía el patrón de ocupación, en donde la selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno de la pluralidad de nodos vecinos. El método incluye además codificar por entropía el patrón de ocupación en función de la una o más probabilidades seleccionadas, usando el uno o más codificadores entrópicos asociados, para producir datos codificados para el flujo de bits.

En otro aspecto, la presente solicitud proporciona un método para decodificar un flujo de bits de datos comprimidos de nube de puntos para producir una nube de puntos reconstruida, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolumenes y que contiene los puntos de la nube de puntos. El método incluye, para un nodo actual asociado con un subvolumen partido en subvolumenes adicionales, correspondiendo cada subvolumen adicional a un nodo hijo del nodo actual, seleccionar una o más probabilidades asociadas con codificadores entrópicos respectivos para decodificar por entropía el patrón de ocupación, en donde la selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación de nodos hijos de al menos uno de la pluralidad de nodos vecinos. El método incluye además decodificar por entropía el flujo de bits en función de la una o más probabilidades seleccionadas, usando el uno o más codificadores entrópicos asociados, para producir un patrón de ocupación reconstruido para la ocupación de señalización de nodo actual de los nodos hijos.

En algunas implementaciones, la selección de una o más probabilidades puede basarse en una configuración de vecino que se determina en función de un estado de ocupación de cada uno de los nodos vecinos del nodo actual.

En algunas implementaciones, un nodo vecino del nodo actual puede considerarse ocupado con el propósito de determinar la configuración de vecino si sus datos de ocupación indican que está ocupado y los datos de ocupación para sus nodos hijos indican que al menos uno de sus nodos hijos ocupados es vecino al nodo actual.

En algunas implementaciones, un nodo vecino del nodo actual puede considerarse como no ocupado con el propósito de determinar la configuración de vecino si sus datos de ocupación indican que está ocupado y los datos de ocupación para sus nodos hijos indican que ninguno de sus nodos hijos ocupados es vecino al nodo actual. Esto puede corresponder a establecer (deliberadamente/artificialmente) el bit de ocupación para ese nodo vecino en cero en la determinación de la configuración de vecino.

En algunas implementaciones, un nodo vecino del nodo actual puede considerarse ocupado con el propósito de determinar la configuración de vecino si sus datos de ocupación indican que está ocupado y si aún no se ha codificado. Cuando un nodo vecino aún no se ha codificado, el decodificador aún no tiene información sobre la ocupación de sus nodos hijos, por lo tanto, tal información no puede usarse para considerar que el nodo vecino está ocupado o no con el propósito de determinar la configuración de vecino.

En algunas implementaciones, un nodo vecino del nodo actual puede considerarse como no ocupado con el propósito de determinar la configuración de vecino si sus datos de ocupación indican que no está ocupado.

En algunas implementaciones, los nodos vecinos de los nodos actuales pueden ser aquellos nodos que están a la misma profundidad en la estructura de árbol que el nodo actual y cuyos subvolumenes asociados intersecan el subvolumen del nodo actual.

En algunas implementaciones, los nodos hijos vecinos al nodo actual pueden ser aquellos nodos que están a una profundidad más baja en uno en la estructura de árbol que el nodo actual y cuyos subvolumenes asociados intersecan el subvolumen del nodo actual.

En algunas implementaciones, los datos de ocupación para la pluralidad de nodos vecinos pueden incluir estados de ocupación para cada uno de la pluralidad de nodos vecinos.

En algunas implementaciones, la estructura de árbol puede representar un árbol octal.

En algunas implementaciones, el método de codificación puede incluir además codificar un indicador que indica que la una o más probabilidades asociadas con los codificadores entrópicos respectivos para codificar por entropía el patrón de ocupación se han seleccionado en función de los datos de ocupación para la pluralidad de nodos vecinos del nodo actual y en los datos de ocupación de los nodos hijos de al menos uno de la pluralidad de nodos vecinos.

En algunas implementaciones, el método de decodificación puede incluir además decodificar un indicador que indica que la una o más probabilidades asociadas con los codificadores entrópicos respectivos para decodificar por entropía el patrón de ocupación deben seleccionarse en función de los datos de ocupación para la pluralidad de nodos vecinos del nodo actual y en los datos de ocupación de los nodos hijos de al menos uno de la pluralidad de nodos vecinos.

5 En otro aspecto, la presente solicitud proporciona un método para codificar una nube de puntos para generar un flujo de bits de datos comprimidos de nube de puntos, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolúmenes y que contiene los puntos de la nube de puntos, en donde la ocupación de subvolúmenes de un volumen se indica usando una secuencia de bits indicando cada bit de la secuencia de bits la ocupación de un subvolumen respectivo en un orden de exploración dentro del volumen, y en donde un volumen tiene una pluralidad de volúmenes vecinos, siendo un patrón de ocupación de los volúmenes vecinos una configuración de vecino. El método incluye, para al menos un bit en la secuencia de bits del volumen, determinar que se cumple una condición de reducción de contexto y, sobre esa base, seleccionar un conjunto de contexto reducido que contiene menos contextos que el producto de un recuento de configuraciones de vecino y un número de bits previamente codificados en la secuencia; seleccionar, para codificar el al menos un bit, un contexto del conjunto de contexto reducido en función de un estado de ocupación de al menos algunos de los volúmenes vecinos y al menos un bit previamente codificado de la secuencia de bits; codificar por entropía el al menos un bit en función del contexto seleccionado usando un codificador entrópico binario para producir datos codificados para el flujo de bits; y actualizar el contexto seleccionado.

20 En otro aspecto, la presente solicitud proporciona un método para decodificar un flujo de bits de datos comprimidos de nube de puntos para producir una nube de puntos reconstruida, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolúmenes y que contiene los puntos de la nube de puntos, en donde la ocupación de subvolúmenes de un volumen se indica usando una secuencia de bits con cada bit de la secuencia de bits que indica la ocupación de un subvolumen respectivo en un orden de exploración dentro del volumen, y en donde un volumen tiene una pluralidad de volúmenes vecinos, siendo un patrón de ocupación de los volúmenes vecinos una configuración de vecino.

El método de decodificación incluye, para al menos un bit en la secuencia de bits del volumen, determinar que se cumple una condición de reducción de contexto y, sobre esa base, seleccionar un conjunto de contexto reducido que contiene menos contextos que el producto de un recuento de configuraciones de vecino y un número de bits previamente codificados en la secuencia; seleccionar, para codificar el al menos un bit, un contexto del conjunto de contexto reducido en función de un estado de ocupación de al menos algunos de los volúmenes vecinos y al menos un bit previamente codificado de la secuencia de bits; decodificar por entropía el al menos un bit en función del contexto seleccionado usando un decodificador entrópico binario para producir un bit reconstruido a partir del flujo de bits; y actualizar el contexto seleccionado.

En algunas implementaciones, la condición de reducción de contexto puede incluir determinar que uno o más bits de ocupación previamente codificados se asocian con uno o más subvolúmenes respectivos colocados entre el subvolumen asociado con el al menos un bit y el uno o más de los volúmenes vecinos. En algunos casos, esto puede incluir determinar que cuatro subvolúmenes asociados con bits previamente codificados comparten una cara con un volumen de vecino particular.

En algunas implementaciones, la condición de reducción de contexto puede incluir determinar que al menos cuatro bits de la secuencia de bits se han codificado previamente.

En algunas implementaciones, determinar que se cumple la condición de reducción de contexto puede incluir determinar que el patrón de ocupación de los volúmenes vecinos indica que la pluralidad de volúmenes vecinos no está ocupada. En algunos de esos casos, el conjunto de contexto reducido seleccionado puede incluir un número de contextos correspondientes al número de bits previamente codificados en la secuencia de bits y, opcionalmente, seleccionar el contexto puede incluir seleccionar el contexto en función de una suma de bits previamente codificados en la secuencia de bits.

En algunas implementaciones, la condición de reducción de contexto puede incluir determinar que al menos un número umbral de bits en la secuencia de bits se ha codificado previamente, y el conjunto de contexto reducido puede incluir una tabla de consulta que correlaciona cada combinación posible de configuración de vecino y patrón de bits previamente codificados en la secuencia de bits con los menos contextos. En algunos ejemplos, la tabla de consulta puede generarse en función de una agrupación iterativa de contextos disponibles en una pluralidad de clases en función de la determinación de que una medición de distancia entre pares respectivos de contextos disponibles es menor que un valor umbral, y cada clase en la pluralidad de clases puede incluir un contexto respectivo en el conjunto más pequeño, y puede haber un contexto disponible para cada una de la posible combinación de configuración de vecino y patrón de bits previamente codificados en la secuencia de bits.

En algunas implementaciones, al menos algunos de los volúmenes vecinos son volúmenes vecinos que comparten al menos una cara con el volumen.

En un aspecto adicional, la presente solicitud describe codificadores y decodificadores configurados para implementar tales métodos de codificación y decodificación.

En otro aspecto más, la presente solicitud describe medios legibles por ordenador no transitorios que almacenan instrucciones de programa ejecutables por ordenador que, cuando se ejecutan, hacen que uno o más procesadores realicen los métodos descritos de codificación y/o decodificación.

En otro aspecto más, la presente solicitud describe una señal legible por ordenador que contiene instrucciones de programa que, cuando se ejecutan por un ordenador, hacen que el ordenador realice los métodos descritos de codificación y/o decodificación.

La presente solicitud describe además aplicaciones implementadas por ordenador, incluyendo aplicación de topografía, aplicaciones de cartografía, aplicaciones de la industria del automóvil, aplicaciones de conducción autónoma, aplicaciones de realidad virtual y aplicaciones de heredación cultural, etc. Estas aplicaciones implementadas por ordenador incluyen procesos de recepción de un flujo de datos o archivo de datos, desempaquetado del flujo de datos o archivo de datos para obtener un flujo de bits de datos comprimidos en la nube de puntos y decodificación del flujo de bits como se describe en los aspectos anteriores y sus implementaciones. De este modo, estas aplicaciones implementadas por ordenador hacen uso de una técnica de compresión de nube de puntos según aspectos y sus implementaciones descritas a lo largo de la presente solicitud.

La presente solicitud describe además métodos de codificación y decodificación de nubes de puntos, y codificadores y decodificadores para codificar y decodificar nubes de puntos. En algunas implementaciones, una unidad de recepción recibe datos multiplexados que se obtienen multiplexando datos en la nube de puntos codificados con otros tipos de datos codificados tales como metadatos, imagen, vídeo, audio y/o gráficos. La unidad de recepción comprende una unidad de demultiplexación para separar los datos multiplexados en datos de puntos codificados y otros datos codificados, y al menos una unidad de decodificación (o decodificador) para decodificar los datos en la nube de puntos codificados. En algunas otras implementaciones, una unidad de emisión emite datos multiplexados que se obtienen multiplexando datos en la nube de puntos codificados con otros tipos de datos codificados tales como metadatos, imagen, vídeo, audio y/o gráficos. La unidad de emisión comprende al menos una unidad de codificación (o codificador) para codificar los datos en la nube de puntos, y una unidad de multiplexación para combinar datos en la nube de puntos codificados y otros datos codificados en los datos multiplexados.

Otros aspectos y características de la presente solicitud serán entendidos por los expertos en la técnica a partir de una revisión de la siguiente descripción de ejemplos junto con las figuras adjuntas.

Cualquier característica descrita en relación con un aspecto o realización de la invención también puede usarse con respecto a uno o más aspectos/realizaciones distintos. Estos y otros aspectos de la presente invención serán evidentes a partir de las realizaciones, y se aclararán con referencia a estas, descritas en esta memoria.

En los momentos de la descripción a continuación, los términos "nodo", "volumen" y "subvolumen" pueden usarse indistintamente. Se apreciará que un nodo se asocia con un volumen o subvolumen. El nodo es un punto particular en el árbol que puede ser un nodo interno o un nodo hoja. El volumen o subvolumen es el espacio físico delimitado que representa el nodo. El término "volumen" puede, en algunos casos, usarse para referirse al espacio delimitado más grande definido para contener la nube de puntos. Un volumen puede dividirse recursivamente en subvolúmenes con el fin de construir una estructura de árbol de nodos interconectados para codificar los datos de nube de puntos.

En la presente solicitud, el término "y/o" pretende cubrir todas las combinaciones y subcombinaciones posibles de los elementos enumerados, incluyendo uno cualquiera de los elementos enumerados solo, cualquier subcombinación o todos los elementos, y sin excluir necesariamente elementos adicionales.

En la presente solicitud, la frase "al menos uno de... o..." pretende cubrir uno cualquiera o más de los elementos enumerados, incluyendo uno cualquiera de los elementos enumerados solo, cualquier subcombinación, o todos los elementos, sin excluir necesariamente ningún elemento adicional, y sin requerir necesariamente todos los elementos.

Una nube de puntos es un conjunto de puntos en un sistema de coordenadas tridimensional. Los puntos están destinados a menudo a representar la superficie externa de uno o más objetos. Cada punto tiene una ubicación (posición) en el sistema de coordenadas tridimensionales. La posición puede representarse por tres coordenadas (X, Y, Z), que pueden ser cartesianas o cualquier otro sistema de coordenadas. Los puntos pueden tener otros atributos asociados, tales como color, que también puede ser un valor de tres componentes en algunos casos, tales como R, G, B o Y, Cb, Cr. Otros atributos asociados pueden incluir transparencia, reflectancia, un vector normal, etc., dependiendo de la aplicación deseada para los datos de nube de puntos.

Las nubes de puntos pueden ser estáticas o dinámicas. Por ejemplo, un escaneo o mapeo detallado de un objeto o topografía pueden ser datos estáticos de la nube de puntos. La exploración basada en LiDAR de un entorno con fines de visión de máquina puede ser dinámica porque la nube de puntos (al menos potencialmente) cambia con el tiempo, por ejemplo, con cada exploración sucesiva de un volumen. La nube de puntos dinámica es, por lo tanto, una secuencia ordenada en el tiempo de nubes de puntos.

Los datos de la nube de puntos pueden usarse en varias aplicaciones, incluyendo conservación (exploración de objetos históricos o culturales), mapeo, visión de máquina (tal como coches autónomos o semiautónomos) y sistemas de realidad virtual, para dar algunos ejemplos. Los datos dinámicos de la nube de puntos para aplicaciones como la visión de máquina pueden ser bastante diferentes de los datos estáticos de la nube de puntos como los de conservación. La visión de automoción, por ejemplo, típicamente implica nubes de puntos altamente dinámicas, no coloreadas, de resolución relativamente pequeña obtenidas a través de sensores LiDAR (o similares) con una alta frecuencia de captura. El objetivo de tales nubes de puntos no es para el consumo o la visión humana sino más bien para la detección/clasificación de objetos de máquina en un proceso de decisión. Como ejemplo, las tramas LiDAR típicas contienen del orden de decenas de miles de puntos, mientras que las aplicaciones de realidad virtual de alta calidad requieren varios millones de puntos. Puede esperarse que exista una demanda de datos de mayor resolución a lo largo del tiempo a medida que aumente la velocidad computacional y se encuentren nuevas aplicaciones.

Aunque los datos de nube de puntos son útiles, una falta de compresión efectiva y eficiente, es decir, los procesos de codificación y decodificación pueden obstaculizar la adopción y el despliegue. Un reto particular en la codificación de nubes de puntos que no surge en el caso de otra compresión de datos, como audio o vídeo, es la codificación de la geometría de la nube de puntos. Las nubes de puntos tienden a estar escasamente pobladas, lo que hace que la codificación eficiente de la ubicación de los puntos sea mucho más desafiante.

Uno de los mecanismos más comunes para codificar datos de nube de puntos es a través del uso de estructuras basadas en árboles. En una estructura basada en árbol, el volumen tridimensional delimitador para la nube de puntos se divide recursivamente en subvolúmenes. Los nodos del árbol corresponden a subvolúmenes. La decisión de si dividir o no adicionalmente un subvolumen puede basarse en la resolución del árbol y/o si hay algún punto contenido en el subvolumen. Un nodo hoja puede tener un indicador de ocupación que indica si su subvolumen asociado contiene o no un punto. Los indicadores de partición pueden señalar si un nodo tiene nodos hijos (es decir, si un volumen actual se ha partido aún más en subvolúmenes). Estos indicadores pueden codificarse por entropía en algunos casos y en algunos casos puede usarse codificación predictiva.

Una estructura de árbol usada comúnmente es un árbol octal. En esta estructura, los volúmenes/subvolúmenes son todos cubos y cada partición de un subvolumen da como resultado ocho subvolúmenes/subcubos adicionales. Otra estructura de árbol comúnmente usada es un árbol KD, en donde un volumen (cubo o cuboide rectangular) se divide recursivamente en dos por un plano ortogonal a uno de los ejes. Las árboles octales son un caso especial de árboles KD, donde el volumen se divide por tres planos, siendo cada uno ortogonal a uno de los tres ejes. Ambos ejemplos se refieren a cubos o cuboides rectangulares; sin embargo, la presente solicitud no se limita a tales estructuras de árbol y los volúmenes y subvolúmenes pueden tener otras formas en algunas aplicaciones. La partición de un volumen no es necesariamente en dos subvolúmenes (árbol KD) u ocho subvolúmenes (árbol octal), sino que podría implicar otras particiones, incluyendo la división en formas no rectangulares o que impliquen subvolúmenes no adyacentes.

La presente solicitud puede referirse a árboles octales para facilitar la explicación y debido a que son una estructura de árbol candidata popular para aplicaciones de automoción, pero se entenderá que los métodos y dispositivos descritos en esta memoria pueden implementarse usando otras estructuras de árbol.

Ahora se hace referencia a la Figura 1, que muestra un diagrama de bloques simplificado de un codificador 10 en la nube de puntos según aspectos de la presente solicitud. El codificador de nube de puntos 10 incluye un módulo de construcción de árbol 12 para recibir datos de nube de puntos y producir un árbol (en este ejemplo, un árbol octal) que representa la geometría del espacio volumétrico que contiene la nube de puntos e indica la ubicación o posición de los puntos desde la nube de puntos en esa geometría.

El proceso básico para crear un árbol octal para codificar una nube de puntos puede incluir:

1. Comenzar con un volumen delimitador (cubo) que contiene la nube de puntos en un sistema de coordenadas
2. Partir el volumen en 8 subvolúmenes (ocho subcubos)
3. Para cada subvolumen, marcar el subvolumen con 0 si el subvolumen está vacío, o con 1 si hay al menos un punto en él
4. Para todos los subvolúmenes marcados con 1, repetir (2) el partir esos subvolúmenes, hasta que se alcance una profundidad máxima de partición
5. Para todos los subvolúmenes de hojas (subcubos) de profundidad máxima, marcar el cubo de hojas con 1 si no está vacío, 0 de lo contrario

El proceso anterior podría describirse como un proceso de ocupación-igual-partición, donde la partición implica la ocupación, con la restricción de que hay una profundidad o resolución máxima más allá de la cual no se producirá partición adicional. En este caso, un único indicador señala si un nodo está partido y, por lo tanto, si está ocupado por al menos un punto, y viceversa. A la profundidad máxima, el indicador señala la ocupación, sin que sea posible una partición adicional.

En algunas implementaciones, la partición y la ocupación son independientes de manera que un nodo puede estar ocupado y puede o no estar partido. Existen dos variaciones de esta implementación:

1. Partido entonces ocupado. Un indicador de señal indica si un nodo está partido. Si se parte, entonces el nodo debe contener un punto, es decir, la partición implica la ocupación. En caso contrario, si el nodo no debe partirse, entonces otro indicador de ocupación señala si el nodo contiene al menos un punto. Por consiguiente, cuando un nodo no se parte adicionalmente, es decir, es un nodo hoja, el nodo hoja debe tener un indicador de ocupación asociado para indicar si contiene algún punto.
2. Ocupado-entonces-partido. Un único indicador indica si el nodo está ocupado. Si no está ocupado, entonces no se produce partición. Si está ocupado, entonces se codifica un indicador de partición para indicar si el nodo está partido o no adicionalmente.

Independientemente de cuál de los procesos descritos anteriormente se use para construir el árbol, puede atravesarse en un orden predefinido (primero en anchura o primero en profundidad, y según un patrón/orden de exploración dentro de cada subvolumen dividido) para producir una secuencia de bits a partir de los indicadores (indicadores de ocupación y/o partición). Esto puede denominarse serialización o binarización del árbol. Como se muestra en la Figura 1, en este ejemplo, el codificador de nube de puntos 10 incluye un binarizador 14 para binarizar el árbol octal para producir un flujo de bits de datos binarizados que representan el árbol.

Esta secuencia de bits puede codificarse entonces usando un codificador entrópico 16 para producir un flujo de bits comprimido. El codificador entrópico 16 puede codificar la secuencia de bits usando un modelo de contexto 18 que especifica probabilidades para codificar bits en función de una determinación de contexto por el codificador entrópico 16. El modelo de contexto 18 puede actualizarse de manera adaptativa después de la codificación de cada bit o conjunto definido de bits. El codificador entrópico 16 puede, en algunos casos, ser un codificador aritmético binario. El codificador aritmético binario puede, en algunas implementaciones, emplear codificación aritmética binaria adaptativa al contexto (CABAC). En algunas implementaciones, se pueden usar codificadores distintos de los codificadores aritméticos.

En algunos casos, el codificador entrópico 16 puede no ser un codificador binario, sino que en su lugar puede operar sobre datos no binarios. Los datos de árbol octal de salida desde el módulo de construcción de árbol 12 pueden no ser evaluados en forma binaria sino que en su lugar pueden ser codificados como datos no binarios. Por ejemplo, en el caso de un árbol octal, los ocho indicadores dentro de un subvolumen (por ejemplo, indicadores de ocupación) en su orden de exploración pueden considerarse un 2^8-1 número de bits (por ejemplo, un número entero que tiene un valor entre 1 y 255, ya que el valor 0 no es posible para un subvolumen partido, es decir, no habría sido partida si estuviera totalmente desocupada). Este número puede ser codificado por el codificador entrópico utilizando un codificador aritmético de múltiples símbolos en algunas implementaciones. Dentro de un subvolumen, p. ej. un cubo, la secuencia de indicadores que define este entero puede denominarse "patrón".

Al igual que con la codificación de vídeo o de imagen, la codificación de nube de puntos puede incluir operaciones predictivas en donde se realizan esfuerzos para predecir el patrón para un subvolumen. Las predicciones pueden ser espaciales (dependientes de subvolúmenes previamente codificados en la misma nube de puntos) o temporales (dependientes de nubes de puntos previamente codificadas en una secuencia ordenada en el tiempo de nubes de puntos).

En la Figura 2 se muestra un diagrama de bloques de un decodificador de nube de puntos 50 de ejemplo que corresponde al codificador 10. El decodificador de nube de puntos 50 incluye un decodificador entrópico 52 que usa el mismo modelo de contexto 54 usado por el codificador 10. El decodificador entrópico 52 recibe el flujo de bits de entrada de datos comprimidos y decodifica por entropía los datos para producir una secuencia de salida de bits descomprimidos. La secuencia se convierte entonces en datos de nube de puntos reconstruidos por un reconstructor de árboles 56. El reconstructor de árboles 56 reconstruye la estructura de árbol a partir de los datos descomprimidos y el conocimiento del orden de exploración en donde se binarizaron los datos de árbol. El reconstructor de árboles 56 es capaz de reconstruir la ubicación de los puntos a partir de la nube de puntos (sujeto a la resolución de la codificación de árbol).

Un subvolumen parcial 100 de ejemplo se muestra en la Figura 3. En este ejemplo, se muestra un subvolumen 100 en dos dimensiones para facilitar la ilustración, y el tamaño del subvolumen 100 es 16×16 . Se observará que el subvolumen se ha dividido en cuatro subcuadrados 8×8 , y dos de ellos se han subdividido adicionalmente en subcuadrados 4×4 , tres de los cuales se dividen adicionalmente en subcuadrados 2×2 , y uno de los subcuadrados 2×2 se divide entonces en cuadrados 1×1 . Los cuadrados 1×1 son la profundidad máxima del árbol y representan la resolución más fina para los datos de puntos de posición. Los puntos de la nube de puntos se muestran como puntos en la figura.

La estructura del árbol 102 se muestra a la derecha del subvolumen 100. La secuencia de indicadores de partición 104 y la secuencia correspondiente de indicadores de ocupación 106, obtenida en un primer orden de exploración de amplitud predefinido, se muestra a la derecha del árbol 102. Se observará que en este ejemplo ilustrativo, hay un indicador de ocupación para cada subvolumen (nodo) que no está partido, es decir, que tiene un indicador de partición asociado puesto a cero. Estas secuencias pueden ser codificadas por entropía.

Otro ejemplo, que emplea una condición de partición $\equiv$ ocupada, se muestra en la Figura 4. La Figura 4 ilustra la partición y codificación recursivas de un árbol octal 150. En la figura sólo se muestra una parte del árbol octal 150. Se

muestra una FIFO 152 como el procesamiento de los nodos para la partición para ilustrar la primera naturaleza de amplitud del presente proceso. La memoria FIFO 152 tiene como salida un nodo ocupado 154 que fue puesto en cola en la memoria FIFO 152 para partición adicional después del procesamiento de su nodo padre 156. El constructor de árboles parte el subvolumen asociado con el nodo ocupado 154 en ocho subvolúmenes (cubos) y determina su ocupación. La ocupación puede indicarse mediante un indicador de ocupación para cada subvolumen. En un orden de exploración prescrito, los indicadores pueden denominarse patrón de ocupación para el nodo 154. El patrón puede especificarse por el número entero que representa la secuencia de indicadores de ocupación asociados con los subvolúmenes en el orden de exploración predefinido. En el caso de un árbol octal, el patrón es un número entero en el intervalo [1, 255],

El codificador entrópico codifica entonces ese patrón usando un codificador aritmético no binario basado en probabilidades especificadas por el modelo de contexto. En este ejemplo, las probabilidades pueden ser una distribución de patrones basada en un modelo de distribución inicial y actualizada de forma adaptativa. En una implementación, la distribución de patrones es efectivamente un contador del número de veces que se ha encontrado cada patrón (número entero de 1 a 255) durante la codificación. La distribución de patrones puede actualizarse después de codificar cada subvolumen. La distribución de patrones puede normalizarse, según sea necesario, ya que la frecuencia relativa de los patrones es dependiente de la evaluación de probabilidad y no del recuento absoluto.

En función del patrón, aquellos nodos hijos que están ocupados (p. ej. hacer que un indicador = 1) se insertan entonces en la FIFO 152 para partición adicional a su vez (siempre que los nodos no sean una profundidad máxima del árbol).

Se hace referencia ahora a la Figura 5, que muestra un cubo 180 de ejemplo de un árbol octal. El cubo 180 se subdivide en ocho subcubos. El orden de exploración para leer los indicadores da lugar a una cadena de ocho bits, que puede leerse como un entero [1, 255] en binario. En función del orden de exploración y la posición de bit resultante de cada indicador de subcubo en la cadena, los subcubos tienen los valores mostrados en la Figura 5. El orden de exploración puede ser cualquier secuencia de los subcubos, siempre que tanto el codificador como el decodificador usen el mismo orden de exploración.

Como ejemplo, la Figura 6 muestra el cubo 180 en donde están ocupados los cuatro subcubos "delanteros". Esto correspondería al patrón 85, sobre la base de que los subcubos ocupados son cubos 1+4+16+64. El número de patrón entero especifica el patrón de ocupación en los subcubos.

Una representación de árbol octal, o más generalmente cualquier representación de árbol, es eficiente en la representación de puntos con una correlación espacial porque los árboles tienden a factorizar los bits de orden superior de las coordenadas de punto. Para un árbol octal, cada nivel de profundidad refina las coordenadas de puntos dentro de un subvolumen en un bit para cada componente a un coste de ocho bits por refinamiento. Se obtiene una compresión adicional al codificar por entropía la información partida, es decir, patrón, asociado a cada nodo árbol. Esta compresión adicional es posible porque la distribución del patrón no es uniforme, siendo la falta de uniformidad otra consecuencia de la correlación.

Una ineficiencia potencial en los sistemas actuales es que la distribución de patrones (por ejemplo, el histograma de los números de patrones vistos en nodos previamente codificados del árbol) se desarrolla durante el curso de la codificación de la nube de puntos. En algunos casos, la distribución de patrones puede inicializarse como equiprobable, o puede inicializarse a alguna otra distribución predeterminada; pero el uso de una distribución de patrones significa que el modelo de contexto no tiene en cuenta, o explota, la correlación geométrica local.

En la solicitud de patente europea n.º 18305037.6 los presentes solicitantes describieron métodos y dispositivos para seleccionar entre distribuciones de patrones disponibles que van a usarse en la codificación de un patrón de ocupación de un nodo particular en función de alguna información de ocupación desde nodos previamente codificados cerca del nodo particular. En una implementación de ejemplo, la información de ocupación se obtiene a partir del patrón de ocupación del padre al nodo particular. En otra implementación de ejemplo, la información de ocupación se obtiene de uno o más nodos vecinos del nodo particular. El contenido de la solicitud de patente europea n.º 18305037.6 se incorporan en la presente memoria por referencia.

Ahora se hace referencia a la Figura 7, que muestra, en forma de diagrama de flujo, un método 200 de ejemplo para codificar una nube de puntos. El método 200 en este ejemplo implica la partición recursiva de nodos ocupados (subvolúmenes) y un primer cruce de amplitud del árbol para codificación.

En la operación 202, el codificador determina el patrón de ocupación para el nodo actual. El nodo actual es un nodo ocupado que se ha partido en ocho nodos hijos, correspondiendo cada uno a un subcubo respectivo. El patrón de ocupación para el nodo actual especifica la ocupación de los ocho nodos hijos en el orden de exploración. Como se ha descrito anteriormente, este patrón de ocupación puede indicarse usando un número entero entre 1 y 255, por ejemplo, una cadena binaria de ocho bits.

En la operación 204, el codificador selecciona una distribución de probabilidad entre un conjunto de distribuciones de probabilidad. La selección de la distribución de probabilidad se basa en alguna información de ocupación de nodos cercanos previamente codificados, es decir, al menos un nodo que es vecino al nodo actual. Dos nodos son vecinos, en algunas realizaciones, si se asocian con respectivos subvolúmenes que comparten al menos una cara. En una

definición más amplia, los nodos son vecinos si comparten al menos un borde. En una definición más amplia, dos nodos son vecinos si comparten al menos un vértice. El patrón padre dentro del que el nodo actual es un nodo hijo, proporciona datos de ocupación para el nodo actual y los siete nodos hermanos al nodo actual. En algunas implementaciones, la información de ocupación es el patrón padre. En algunas implementaciones, la información de ocupación son datos de ocupación para un conjunto de nodos vecinos que incluyen nodos en el mismo nivel de profundidad del árbol que el nodo actual, pero que tienen un nodo padre diferente. En algunos casos, son posibles combinaciones de estos. Por ejemplo, un conjunto de nodos vecinos puede incluir algunos nodos hermanos y algunos nodos no hermanos.

Una vez que se ha seleccionado la distribución de probabilidad, el codificador codifica por entropía entonces el patrón de ocupación para el nodo actual usando la distribución de probabilidad seleccionada, tal como se indica mediante la operación 206. Entonces actualiza la distribución de probabilidad seleccionada en la operación 208 en función del patrón de ocupación, por ejemplo, puede incrementar el recuento correspondiente a ese patrón de ocupación. En la operación 210, el codificador evalúa si hay otros nodos para codificar y, si es así, vuelve a la operación 202 para codificar el siguiente nodo.

La selección de distribución de probabilidad en la operación 204 se basa en datos de ocupación para nodos cercanos previamente codificados. Esto permite que tanto el codificador como el decodificador realicen independientemente la misma selección. Para la siguiente discusión de la selección de distribución de probabilidad, se hará referencia a la Figura 8, que ilustra esquemáticamente un árbol octal parcial 300, que incluye un nodo actual 302. El nodo actual 302 es un nodo ocupado y está siendo evaluado para codificación. El nodo actual 302 es uno de ocho hijos de un nodo padre 306, que a su vez es un hijo de un nodo abuelo (no mostrado). El nodo actual 302 se divide en ocho nodos hijos 304. El patrón de ocupación para el nodo actual 302 se basa en la ocupación de los nodos hijos 304. Por ejemplo, como se ilustra, usando la convención de que un punto negro es un nodo ocupado, el patrón de ocupación puede ser 00110010, es decir, 50.

El nodo actual 302 tiene nodos hermanos 308 que tienen el mismo nodo padre 306. El patrón padre es el patrón de ocupación para el nodo padre 306, que como se ilustra sería 00110000, es decir, patrón 48. El patrón padre puede servir como base para seleccionar una distribución de probabilidad adecuada para codificar por entropía el patrón de ocupación para el nodo actual.

La Figura 9 ilustra un conjunto de vecinos que rodean un nodo actual, donde vecino se define como nodos que comparten una cara. En este ejemplo, los nodos/subvolumenes son cubos y el cubo en el centro de la imagen tiene seis vecinos, uno para cada cara. En un árbol octal, se apreciará que los vecinos al nodo actual incluirán tres nodos hermanos. También incluirá tres nodos que no tienen el mismo nodo padre. Por consiguiente, los datos de ocupación para algunos de los nodos vecinos estarán disponibles porque son hermanos, pero los datos de ocupación para algunos nodos vecinos pueden o no estar disponibles, dependiendo de si esos nodos se codificaron previamente. Se puede aplicar manejo especial para tratar a los vecinos ausentes. En algunas implementaciones, se puede suponer que el vecino que falta está ocupado o se puede suponer que no está ocupado. Se apreciará que la definición de vecino puede ensancharse para incluir nodos vecinos en función de un borde compartido o en función de un vértice compartido para incluir subvolumenes adyacentes adicionales en la evaluación.

Se apreciará que los procesos anteriores buscan la ocupación de nodos cercanos en un intento de determinar la probabilidad de ocupación del nodo actual 302 para seleccionar contexto(s) más adecuado(s) y usar probabilidades más precisas para codificar por entropía los datos de ocupación del nodo actual 302. Se entenderá que el estado de ocupación de nodos vecinos que comparten una cara con el nodo actual 302 puede ser una evaluación más precisa de si es probable que el nodo actual 302 esté aislado o no en función de esa evaluación en el estado de ocupación de nodos hermanos, tres de los cuales solo compartirán un borde y uno de los cuales solo compartirá un vértice (en el caso de un árbol octal). Sin embargo, la evaluación del estado de ocupación de hermanos tiene la ventaja de ser modular en que todos los datos relevantes para la evaluación son parte del nodo padre, lo que significa que tiene una huella de memoria más pequeña para la implementación, mientras que la evaluación del estado de ocupación del vecino implica almacenar temporalmente datos de ocupación del árbol en caso de que se necesite cuando se determina el estado de ocupación del vecino en relación con la codificación de un futuro nodo cercano.

La ocupación de los vecinos puede leerse en un orden de exploración que asigna efectivamente un valor a cada vecino, muy similar a como se ha descrito anteriormente con respecto a los patrones de ocupación. Como se ilustra, los nodos vecinos toman efectivamente valores de 1, 2, 4, 8, 16 o 32, y por lo tanto hay 64 (0 a 63) posibles configuraciones de ocupación de vecinos. Este valor puede denominarse en esta memoria "configuración de vecino". Como ejemplo, la Figura 10 ilustra un ejemplo de configuración de vecino 15, en donde los vecinos 1, 2, 4 y 8 están ocupados y los vecinos 16 y 32 están vacíos.

En algunos casos, los dos criterios anteriores (patrón padre y configuración de vecino) pueden aplicarse o pueden seleccionarse entre ellos. Por ejemplo, si los vecinos están disponibles, entonces la selección de distribución de probabilidad puede realizarse en función de los nodos vecinos; sin embargo, si uno o más de los vecinos no están disponibles porque proceden de nodos todavía no codificados, entonces la selección de distribución de probabilidad puede volver a un análisis en función de nodos hermanos (patrón padre).

En otra realización más, la selección de distribución de probabilidad puede basarse alternativamente, o adicionalmente, en el patrón de abuelo. En otras palabras, la selección de distribución de probabilidad puede basarse en el estado de ocupación de los nodos no centrales que son hermanos al nodo padre 306.

5 En otra implementación adicional, pueden tenerse en cuenta evaluaciones adicionales o alternativas en la selección de distribución de probabilidad. Por ejemplo, la selección de distribución de probabilidad puede buscar el estado de ocupación de nodos vecinos al nodo padre, o nodos vecinos al nodo abuelo.

Dos o más de los criterios anteriores para evaluar el estado de ocupación local pueden usarse en combinación en algunas implementaciones.

10 En el caso de un codificador entrópico no binario, los datos de ocupación para el nodo actual pueden codificarse seleccionando una distribución de probabilidad. La distribución de probabilidad contiene un número de probabilidades correspondiente al número de posibles patrones de ocupación para el nodo actual. Por ejemplo, en el caso de codificar el patrón de ocupación de un árbol octal, hay $2^8-1 = 255$ patrones posibles, lo que significa que cada distribución de probabilidad incluye 255 probabilidades. En algunas realizaciones, el número de distribuciones de probabilidad puede ser igual al número de posibles resultados de ocupación en los criterios de selección, es decir, usando datos de ocupación
15 vecinos, hermanos y/o padres. Por ejemplo, en un caso en donde se usa un patrón padre para un árbol octal como criterio de selección para determinar la distribución de probabilidad a usar, habría 255 distribuciones de probabilidad que implicasen 255 probabilidades cada una. En el caso de la configuración de vecino, si se define vecino como que comparte una cara, habría 64 distribuciones de probabilidad con cada distribución que contiene 255 probabilidades.

20 Se entenderá que demasiadas distribuciones pueden dar como resultado una adaptación lenta debido a la escasez de datos, es decir, dilución de contexto. Por consiguiente, en algunas realizaciones, se pueden agrupar patrones similares para usar la misma distribución de probabilidad. Por ejemplo, se pueden usar distribuciones separadas para patrones correspondientes a totalmente ocupados, orientados verticalmente, orientados horizontalmente, en su mayoría vacíos, y luego todos los demás casos. Esto podría reducir el número de distribuciones de probabilidad a aproximadamente cinco. Se apreciará que se podrían formar diferentes agrupaciones de patrones para dar como
25 resultado un número diferente de distribuciones de probabilidad.

Ahora se hace referencia a la Figura 11, que muestra esquemáticamente una realización ilustrativa de un proceso 400 de codificación entrópica de nube de puntos usando contexto dependiente del patrón padre. En este ejemplo, un nodo actual 402 se ha partido en ocho nodos hijos y su patrón de ocupación 404 debe codificarse usando un codificador entrópico no binario 406. El codificador entrópico no binario 406 usa una distribución de probabilidad seleccionada de
30 una de seis posibles distribuciones de probabilidad 408. La selección se basa en el patrón padre, es decir, la selección se basa en la información de ocupación desde el nodo padre al nodo actual 402.

El patrón padre se identifica por un número entero entre 1 y 255.

La selección de la distribución de probabilidad puede ser un árbol de decisión que evalúa si el patrón corresponde a un nodo completo (por ejemplo, patrón = 255), una estructura horizontal (por ejemplo, patrón = 170 u 85; suponiendo que el eje Z es vertical), una estructura vertical (por ejemplo, patrón = 3, 12, 48, 192), una distribución escasamente poblada (p. ej. patrón = 1, 2, 4, 8, 16, 32, 64 o 128; es decir, ninguno de los nodos hermanos está ocupado), una distribución semiescasamente poblada (número total de nodos ocupados entre el nodo actual y los nodos hermanos ≤ 3), y todos los demás casos. Los patrones de ejemplo indicados para las diferentes categorías son meramente
40 ejemplos. Por ejemplo, la categoría "horizontal" puede incluir patrones que implican dos o tres cubos ocupados en el mismo nivel horizontal. La categoría "vertical" puede incluir patrones que implican tres o cuatro cubos ocupados en una disposición similar a una pared. También se apreciará que se pueden usar gradaciones más finas. Por ejemplo, la categoría "horizontal" puede subdividirse además en horizontal en la parte superior del cubo y horizontal en la parte inferior del cubo con diferentes distribuciones de probabilidad para cada uno. Pueden hacerse y asignarse otros agrupamientos de patrones de ocupación que tienen alguna correlación a una distribución de probabilidad
45 correspondiente. Una discusión adicional con respecto a la agrupación de patrones en el contexto de configuraciones de vecino, y la invariancia entre configuraciones de vecino se establece más adelante.

La Figura 12 muestra una realización ilustrativa de un proceso de codificación entrópica 500 en la nube de puntos usando contexto dependiente de la configuración de vecino. Este ejemplo supone la definición de vecina y numeración de configuración de vecino usadas anteriormente en conexión con la Figura 9. Este ejemplo también supone que cada
50 configuración de vecino tiene una distribución de probabilidad dedicada, lo que significa que hay 64 distribuciones de probabilidad diferentes. Un nodo actual 502 tiene un patrón de ocupación 504 que se va a codificar. La distribución de probabilidad se selecciona en función de los nodos vecinos al nodo actual 502. Es decir, la configuración de vecino NC en [0, 63] se encuentra y se usa para seleccionar la distribución de probabilidad asociada.

Se apreciará que en algunas realizaciones, las configuraciones de vecino pueden agruparse de manera que más de
55 una configuración de vecino usa la misma distribución de probabilidad en función de similitudes en los patrones. En algunas realizaciones, el proceso puede usar una disposición diferente de vecinos para la contextualización (selección) de las distribuciones. Pueden añadirse vecinos adicionales tales como los ocho vecinos diagonalmente adyacentes en los tres ejes, o los doce diagonalmente adyacentes en dos ejes. Las realizaciones que evitan vecinos particulares

también pueden usarse, por ejemplo, para evitar usar vecinos que introduzcan dependencias adicionales en una exploración de primera profundidad, o solo introducir dependencias en ejes particulares para reducir el estado de códec para árboles grandes.

5 En este ejemplo, el caso de $NC = 0$ se maneja de una manera específica. Si no hay vecinos que estén ocupados, puede indicar que el nodo actual 502 está aislado. Por consiguiente, el proceso 500 comprueba además cuántos de los nodos hijos están ocupados para el nodo 502 actual. Si sólo está ocupado un nodo hijo, es decir, NumberOccupied (NO) es igual a 1, entonces se codifica un indicador que indica que un único nodo hijo está ocupado y el índice al nodo se codifica usando 3 bits. Si más de un nodo hijo está ocupado, entonces el proceso 500 usa la distribución de probabilidad $NC=0$ para codificar el patrón de ocupación.

10 Ahora se hace referencia a la Figura 13, que muestra, en forma de diagrama de flujo, un método 600 de ejemplo para decodificar un flujo de bits de datos de nube de puntos codificados.

15 En la operación 602, el decodificador selecciona una de las distribuciones de probabilidad en función de la información de ocupación de uno o más nodos cerca del nodo actual. Como se ha descrito anteriormente, la información de ocupación puede ser un patrón padre desde el nodo padre al nodo actual, es decir, ocupación del nodo actual y sus hermanos, o puede ser ocupación de nodos vecinos al nodo actual, que puede incluir algunos de los nodos hermanos. En algunas implementaciones puede usarse otra información de ocupación o información adicional.

20 Una vez que se ha seleccionado la distribución de probabilidad, entonces en la operación 604 el decodificador decodifica por entropía una parte del flujo de bits usando la distribución de probabilidad seleccionada para reconstruir el patrón de ocupación para el nodo actual. El patrón de ocupación se usa por el decodificador para reconstruir el árbol para reconstruir los datos de nube de puntos codificados. Una vez que se decodifican los datos de la nube de puntos, pueden ser la salida desde el decodificador para su uso, tal como para representar una vista, segmentación/clasificación u otras aplicaciones.

25 En la operación 606, el decodificador actualiza la distribución de probabilidad en función del patrón de ocupación reconstruido, y luego si hay más nodos para decodificar, entonces se mueve al siguiente nodo en el búfer y vuelve a la operación 602.

30 Las implementaciones de ejemplo de los métodos descritos anteriormente han demostrado proporcionar una mejora de compresión con un aumento insignificante en la complejidad de codificación. La selección basada en vecinos muestra un mejor rendimiento de compresión que la selección basada en patrones padre, aunque tiene una mayor complejidad computacional y uso de memoria. En algunas pruebas, la mejora relativa en bits por punto sobre el modelo de prueba de nube de puntos MPEG está entre el 4 y el 20 %. Se ha observado que inicializar las distribuciones de probabilidad en función de una distribución a la que llegan los datos de prueba conduce a un rendimiento mejorado en comparación con inicializar con una distribución uniforme.

35 Algunos de los ejemplos anteriores se basan en un proceso de codificación de árbol que usa un codificador no binario para señalar un patrón de ocupación. A continuación, se presentan más detalladamente nuevos desarrollos para emplear codificadores entrópicos binarios.

40 En una variación de la selección de distribución de probabilidad basada en vecinos, el número de distribuciones puede reducirse aprovechando la simetría de la vecindad. Permutando la proximidad o permutando la distribución de patrones, configuraciones estructuralmente similares que tienen una línea de simetría pueden reutilizar la misma distribución. En otras palabras, las configuraciones de vecino que pueden usar la misma distribución de patrones pueden agruparse en una clase. Una clase que contiene más de una configuración de vecino puede denominarse en esta memoria "configuración de vecino" en que una de las configuraciones de vecino supone efectivamente otras configuraciones de vecino por medio de reflexión o permutación de esas otras configuraciones.

45 Considérese, como ejemplo, los ocho patrones de esquina $NC \in [21, 22, 25, 26, 37, 38, 41, 42]$, representando cada uno una simetría de un patrón vecino de esquina. Es probable que estos valores de NC estén bien correlacionados con patrones particulares pero diferentes de un nodo. Es además probable que estos patrones correlacionados sigan las mismas simetrías que el patrón vecino. A modo de ejemplo, se puede implementar un método que reutiliza una única distribución para representar múltiples casos de NC permutando las probabilidades de esa distribución.

50 Un codificador deriva el número de patrón de un nodo en función de la ocupación de los nodos hijos. El codificador selecciona una función de distribución y permutación según la configuración de vecino. El codificador reordena las probabilidades contenidas dentro de la distribución según la función de permutación, y posteriormente usa la distribución permutada para codificar aritméticamente el número de patrón. Las actualizaciones de las probabilidades de la distribución permutada por el codificador aritmético se correlacionan de nuevo con la distribución original por medio de una función de permutación inversa.

55 Un decodificador correspondiente selecciona primero la misma función de distribución y permutación según la configuración de vecino. Se produce una distribución permutada de manera idéntica al codificador, usándose la distribución permutada por el decodificador aritmético para decodificar por entropía el número de patrón. Los bits que comprenden el número de patrón se asignan entonces cada uno al hijo correspondiente.

Debe observarse que la misma permutación puede conseguirse sin reordenar los datos de la propia distribución, sino más bien introduciendo un nivel de indirección y usando la función de permutación para permutar la búsqueda de un índice dado en la distribución.

Una realización alternativa considera permutaciones del propio patrón en lugar de la distribución, permitiendo una mezcla antes o después de la codificación/decodificación entrópica, respectivamente. Es probable que dicho método sea más susceptible a una implementación eficiente a través de operaciones de barajado de bits. En este caso, no se realiza reordenamiento de la distribución ni por el codificador ni por el decodificador, sino que se modifica la computación del número de patrón codificado para que sea $pn = \sum_{i=0}^7 2^i c_{\sigma(i)}$ donde c_i es el i -ésimo estado de

ocupación del niño, y $\sigma(i)$ es una función de permutación. Una de tales funciones de permutación de ejemplo $\sigma =$

$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 7 & 6 & 5 & 0 & 3 & 2 & 1 \end{pmatrix}$ permite utilizar la distribución para NC = 22 para la de NC = 41. La función de

permutación puede ser usada por un decodificador para derivar un estado de ocupación de un nodo hijo a partir del número de patrón codificado usando $c_i = \lfloor pn/2^{\sigma(i)} \rfloor \bmod 2$.

Los métodos para derivar la permutación requerida pueden basarse en simetrías rotacionales de las configuraciones de vecino, o pueden basarse en reflexiones a lo largo de ejes particulares. Además, no es necesario que la permutación permute todas las posiciones según, por ejemplo, la simetría; en su lugar, se puede usar una permutación parcial. Por ejemplo, cuando se permuta de NC=22 a NC=41, las posiciones en el eje de simetría pueden no

permutarse, lo que conduce a la correlación $\sigma = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 7 & 2 & 5 & 4 & 3 & 6 & 1 \end{pmatrix}$, en donde las posiciones 0, 2, 4, 6 no están permutadas. En otras realizaciones, solo se intercambia el par 1 y 7.

Ejemplos de realizaciones basadas en simetrías y reflexiones rotacionales se proporcionan a continuación para el caso específico de un árbol octal con seis vecinos que comparten una cara común con el cubo actual. Sin pérdida de generalidad, como se muestra en la Figura 16, el eje Z se extiende verticalmente con relación a la dirección de visión de la Figura. Las posiciones relativas de vecinos tales como "por encima" (respectivamente "por debajo") deben entenderse entonces a lo largo del eje Z en la dirección Z creciente (respectivamente decreciente). La misma observación se aplica a izquierda/derecha a lo largo del eje X, y delantero/posterior a lo largo del eje Y.

La Figura 16 muestra tres rotaciones 2102, 2104 y 2106 a lo largo de los ejes Z, Y y X, respectivamente. El ángulo de estas tres rotaciones es de 90 grados, es decir, realizan una rotación a lo largo de su eje respectivo de un cuarto de vuelta.

La Figura 17 muestra clases de invariancia de la configuración de vecino bajo una o varias iteraciones de la rotación 2102 a lo largo del eje Z. Esta invariancia es representativa del mismo comportamiento estadístico de la geometría de la nube de puntos a lo largo de cualquier dirección que pertenezca al plano XY. Esto es particularmente cierto para el caso de uso de un coche que se mueve sobre la superficie de la tierra que se aproxima localmente por el plano XY. Una configuración horizontal es la ocupación dada de los cuatro vecinos (ubicados a la izquierda, derecha, parte delantera y posterior del cubo actual) independientemente de la ocupación del vecino superior (2202) y el vecino inferior (2204). Las cuatro configuraciones horizontales 2206, 2208, 2210 y 2212 pertenecen a la misma clase de invariancia bajo la rotación 2102. De manera similar, las dos configuraciones 2214 y 2216 pertenecen a la misma clase de invariancia. Hay solo seis clases (agrupadas bajo el conjunto de clases 2218) de invariancia bajo la rotación 2102.

Una configuración vertical es la ocupación dada de los dos vecinos 2202 y 2204 independientemente de la ocupación de los cuatro vecinos ubicados a la izquierda, derecha, parte delantera y posterior del cubo actual. Hay cuatro configuraciones verticales posibles como se muestra en la Figura 18. En consecuencia, si se considera la invariancia con respecto a la rotación 2102 a lo largo del eje Z, hay $6 \times 4 = 24$ posibles configuraciones.

La reflexión 2108 a lo largo del eje Z se muestra en la Figura 16. Las configuraciones verticales 2302 y 2304 representadas en la Figura 18 pertenecen a la misma clase de invariancia bajo la reflexión 2108. Hay tres clases (agrupadas bajo el conjunto de clases 2306) de invariancia bajo la reflexión 2108. La invariancia bajo reflexión 2108 significa que las direcciones hacia arriba y hacia abajo se comportan esencialmente igual en términos de estadísticas de geometría de nube de puntos. Es una suposición exacta para un coche en movimiento en una carretera.

Si se asume invariancia tanto bajo la rotación 2102 como bajo la reflexión 2108, hay 18 clases de invariencias, resultantes del producto de los dos conjuntos 2218 y 2306. Estas 18 clases se representan en la Figura 19.

Aplicando una invariancia adicional en las otras dos rotaciones 2104 y 2106, las dos configuraciones 2401 y 2402 pertenecen a la misma clase de invariancia. Además, las dos configuraciones 2411 y 2412, las dos configuraciones 2421 y 2422, las tres configuraciones 2431, 2432 y 2433, las dos configuraciones 2441 y 2442, las dos configuraciones 2451 y 2452, y finalmente las dos configuraciones 2461 y 2462 pertenecen a las mismas clases. En consecuencia, la

invariancia bajo las tres rotaciones (2102, 2104 y 2106) y la reflexión 2108 conduce a 10 clases de invariancia como se muestra en la Figura 20.

A partir de los ejemplos proporcionados anteriormente en esta memoria, suponiendo o no una invariancia bajo tres rotaciones y la reflexión, el número de configuraciones de vecino efectivas, es decir, las clases en donde se puede agrupar la configuración de 64 vecinos, son 64, 24, 18 o 10.

Antes de la codificación entrópica, el patrón experimenta la misma transformación, es decir, rotaciones y reflexión, como la configuración de vecino pertenece a una de las clases invariancia. Esto conserva la consistencia estadística entre la configuración de vecino invariante y el patrón codificado.

También se entenderá que durante el cruce de un árbol, un nodo hijo tendrá ciertos nodos vecinos a la misma profundidad de árbol que se han visitado previamente y pueden usarse causalmente como dependencias. Para estos vecinos de mismo nivel (es decir, al mismo nivel que el nodo hijo), en lugar de consultar el vecino coubicado padre, pueden usarse los vecinos de mismo nivel. Dado que los vecinos del mismo nivel tienen la mitad de las dimensiones del padre, una configuración considera el vecino ocupado si alguno de los cuatro nodos hijos vecinos directamente adyacentes (es decir, los cuatro que comparten una cara con el nodo actual) está ocupado. Por consiguiente, como se describirá con más detalle a continuación, la configuración de vecino de un nodo actual puede determinarse en función de los datos de ocupación de los nodos vecinos del nodo actual y basándose además en datos de ocupación para nodos hijos de al menos uno de los nodos vecinos. Por consiguiente, una o más probabilidades asociadas con los respectivos codificadores entrópicos para codificar por entropía (por ejemplo, codificación entrópica binaria) el patrón de ocupación del nodo actual pueden seleccionarse no solo en función de datos de ocupación para una pluralidad de nodos vecinos (del mismo nivel, es decir, al mismo nivel que el nodo actual) del nodo actual, sino también en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos (del mismo nivel).

Ahora se hace referencia a la Figura 27, que muestra un nodo actual (es decir, su (sub) volumen asociado, o (sub) volumen actual) 4000 y sus seis vecinos 4010, 4020, 4030, 4040, 4050 y 4060. Para el presente ejemplo de un árbol octal, vecinos del nodo actual pueden ser aquellos nodos (al mismo nivel o profundidad del árbol) cuyos volúmenes asociados comparten una cara con el volumen actual. También son factibles otras definiciones de nodos vecinos. Por ejemplo, vecinos del nodo actual pueden ser aquellos nodos (al mismo nivel o profundidad del árbol) cuyos volúmenes asociados comparten un borde (o un vértice) con el volumen actual. En general, independientemente de la estructura de árbol, los nodos vecinos pueden ser aquellos nodos (al mismo nivel o profundidad del árbol) cuyos volúmenes asociados se intersecan con el volumen actual.

En el contexto de la presente solicitud, se entiende que los volúmenes (nodos) que se intersecan entre sí son volúmenes adyacentes (nodos). Por consiguiente, los términos "que tiene una intersección con" y "que es adyacente a" pueden considerarse sinónimos en el contexto de la solicitud.

Notablemente, las expresiones "volumen" y "subvolumen" pueden usarse de manera algo intercambiable, en el sentido de que cada subvolumen es en sí mismo un volumen que puede subdividirse en subvolúmenes. En cualquier caso, se entiende que la relación volumen/subvolumen está clara por la especificación de una relación padre-hijo entre los nodos/volúmenes implicados.

Se supone que el orden de exploración de nodos se realiza en primer lugar en amplitud, en orden X creciente, después orden Y creciente y finalmente en orden Z creciente. Al hacerlo, los tres vecinos con la coordenada X más baja (es decir, el vecino 4010), la coordenada Y más baja (es decir, el vecino 4030) y la coordenada Z más baja (es decir, el vecino 4050) ya están codificados. Por lo tanto, si uno de estos tres vecinos está ocupado, se conoce la configuración de los subvolúmenes ocupados asociados con el vecino ocupado. Aunque el presente ejemplo define un orden de exploración en orden X creciente, entonces aumentar el orden Y y finalmente aumentar el orden Z, también pueden usarse otros primeros órdenes de exploración de amplitud para el presente propósito.

Ahora se hace referencia a la Figura 28, que muestra un volumen actual ejemplar con los tres vecinos ya codificados (es decir, vecinos 4010, 4030 y 4050) ocupados. Los subvolúmenes ocupados del vecino 4010 son subvolúmenes 4011, 4012 y 4013; los subvolúmenes ocupados del vecino 4030 son subvolúmenes 4031, 4032 y 4033; y los subvolúmenes ocupados del vecino 4050 son subvolúmenes 4051 y 4052. En este ejemplo, los tres vecinos ya codificados están ocupados, pero se entiende que en general solo dos o uno de ellos puede estar ocupado realmente, o incluso ninguno de ellos.

El conocimiento de los subvolúmenes ocupados de vecinos ocupados ya codificados puede usarse para refinar el estado de ocupación de un vecino en la computación de la configuración de ocupación del vecino. Ahora se hace referencia a la Figura 29(a), donde el vecino 4010 tiene subvolúmenes ocupados 4014 y 4015, y ninguno de ellos comparte una cara con el volumen actual 4000. En este caso, puede ser ventajoso considerar al vecino 4010 como no ocupado en la computación de la configuración de ocupación del vecino. En la Figura 29(b), al menos uno de los subvolúmenes 4016 y 4017 del vecino 4010 comparte una cara con el volumen actual 4000. En este caso, el vecino 4010 se considera ocupado en la computación de la configuración de ocupación del vecino.

Ahora se hace referencia a la Figura 30, que muestra, en forma de diagrama de flujo, un ejemplo de un método 4100

de codificación de una nube de puntos para generar un flujo de bits de datos comprimidos de nube de puntos. La nube de puntos se define como una estructura de árbol (por ejemplo, un árbol octal) que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representa la geometría de un espacio volumétrico partido recursivamente en subvolúmenes y que contiene los puntos de la nube de puntos. Las operaciones del método 4100 descritas a continuación se realizan cada una para un nodo actual asociado con un (sub)volumen partido en subvolúmenes adicionales, correspondiendo cada subvolumen adicional a un nodo hijo del nodo actual. En la operación 4110, se determina un patrón de ocupación para el nodo actual en función de los estados de ocupación de los nodos hijos. En la operación 4120, se selecciona una o más probabilidades (por ejemplo, contextos) asociadas con los codificadores entrópicos respectivos para codificar por entropía el patrón de ocupación. Esta selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno de la pluralidad de nodos vecinos (posiblemente todos los nodos vecinos). En la operación 4130, el patrón de ocupación se codifica por entropía en función de la una o más probabilidades seleccionadas, usando el uno o más codificadores entrópicos asociados, para producir datos codificados para el flujo de bits.

En alguna implementación, el método 4100 puede incluir además una operación (no mostrada en la Figura 30) de actualizar la una o más probabilidades seleccionadas en función del patrón de ocupación.

El patrón de ocupación del nodo actual puede codificarse por entropía utilizando un codificador entrópico no binario. En este caso, seleccionar una o más probabilidades en la operación 4120 del método 4100 puede corresponder o implicar seleccionar una distribución de probabilidad (y un codificador entrópico no binario asociado) para codificar por entropía el patrón de ocupación. Actualizar la una o más probabilidades seleccionadas puede corresponder o implicar actualizar la distribución de probabilidad seleccionada.

Por otro lado, como se describirá con más detalle a continuación, el patrón de ocupación del nodo actual puede codificarse por entropía usando una cascada de uno o más codificadores entrópicos binarios. Por consiguiente, la operación 4120 del método 4100 puede implicar, para cada bit de la secuencia de bits que representa el patrón de ocupación, seleccionar una probabilidad respectiva (y, en consecuencia, un codificador entrópico asociado) para codificar ese bit. Seleccionar esta probabilidad puede basarse en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos. Además, la selección de esta probabilidad puede basarse en una secuencia parcial de bits ya codificados de la secuencia de bits. Dicho de otro modo, para cada bit de la secuencia de bits, puede seleccionarse un contexto en función de datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos. Además, seleccionar el contexto puede basarse en una secuencia parcial de bits ya codificados de la secuencia de bits. En términos de contexto, se puede decir que la operación 4120 del método 4100 se refiere a seleccionar un contexto para codificar por entropía el patrón de ocupación en función de datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos. A continuación, en algunas implementaciones, este contexto puede actualizarse en función del patrón de ocupación.

Ahora se hace referencia a la Figura 31, que muestra, en forma de diagrama de flujo, un ejemplo de un método 4200 para decodificar un flujo de bits de datos de nube de puntos comprimidos para producir una nube de puntos reconstruida. La nube de puntos se define en una estructura de árbol (por ejemplo, un árbol octal) que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolúmenes y que contiene los puntos de la nube de puntos. Las operaciones del método 4200 descritas a continuación se realizan cada una para un nodo actual asociado a un subvolumen partido en subvolúmenes adicionales, correspondiendo cada subvolumen adicional a un nodo hijo del nodo actual. En la operación 4210, se selecciona una o más probabilidades asociadas con los respectivos codificadores entrópicos para decodificar por entropía el patrón de ocupación. Esta selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación de nodos hijos de al menos uno de la pluralidad de nodos vecinos. En la operación 4220, el flujo de bits se decodifica por entropía en función de la una o más probabilidades seleccionadas, usando el uno o más codificadores entrópicos asociados, para producir un patrón de ocupación reconstruido para la ocupación de señalización de nodo actual de los nodos hijos. En algunas implementaciones, el método 4200 puede incluir además una operación (no mostrada en la Figura 31) de actualizar la una o más probabilidades seleccionadas en función del patrón de ocupación reconstruido.

El patrón de ocupación del nodo actual puede codificarse por entropía utilizando un codificador entrópico no binario. En este caso, seleccionar una o más probabilidades en la operación 4210 del método 4200 puede corresponder o implicar seleccionar una distribución de probabilidad (y un codificador entrópico no binario asociado) para codificar por entropía el patrón de ocupación. Actualizar la una o más probabilidades seleccionadas puede corresponder o implicar actualizar la distribución de probabilidad seleccionada.

Por otro lado, el patrón de ocupación del nodo actual puede codificarse por entropía utilizando una cascada de uno o más codificadores entrópicos binarios. A continuación, de la misma manera que para la codificación, la operación 4210 del método 4200 puede implicar, para cada bit de la secuencia de bits que representa el patrón de ocupación, seleccionar una probabilidad respectiva (y, en consecuencia, un codificador entrópico asociado) para codificar ese bit. Seleccionar esta probabilidad puede basarse en datos de ocupación para una pluralidad de nodos vecinos del nodo

actual y en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos. Además, seleccionar esta probabilidad puede basarse en una secuencia parcial de bits ya codificados en la secuencia de bits. Dicho de otro modo, para cada bit de la secuencia de bits, puede seleccionarse un contexto en función de datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos. Además, seleccionar el contexto puede basarse en una secuencia parcial de bits ya codificados en la secuencia de bits. En términos de contexto, se puede decir que la operación 4210 del método 4200 se refiere a seleccionar un contexto para codificar por entropía el patrón de ocupación en función de datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos. A continuación, en algunas implementaciones, este contexto puede actualizarse en función del patrón de ocupación reconstruido.

En algunas implementaciones del método 4100 y el método 4200, las selecciones respectivas en la operación 4120 y 4210 pueden basarse en la configuración de vecino. Como se ha descrito anteriormente, la configuración de vecino puede determinarse en función de los datos de ocupación para los nodos vecinos (del mismo nivel) del nodo actual. Además, los datos de ocupación para nodos hijos de al menos uno (posiblemente todos) de la pluralidad de nodos vecinos pueden usarse para adaptar el cálculo de la configuración de vecino. En particular, los datos de ocupación para nodos hijos de uno dado de la pluralidad de nodos vecinos pueden usarse para determinar si el nodo vecino dado debe considerarse ocupado o no con el propósito de calcular la configuración de vecino. Un ejemplo de tal adaptación del cálculo de la configuración de vecino se describirá con referencia a la Figura 32.

Ahora se hace referencia a la Figura 32, que muestra, en forma de diagrama de flujo, un método 4300 de ejemplo para decidir la ocupación de vecinos (nodos vecinos) en la computación de la configuración de vecinos. El método se realiza para un volumen actual, para determinar la configuración de vecino del volumen actual. En la operación 4310 se selecciona un vecino del volumen actual. Para cada vecino seleccionado, la ocupación del vecino se comprueba en la operación 4330. Si el vecino no está ocupado (no en la operación 4330), el método avanza a la operación 4340 y el vecino seleccionado se considerará como no ocupado (por ejemplo, que tiene un bit de ocupación que es cero) en la computación de la configuración de ocupación del vecino. Es decir, la ocupación de dicho vecino se toma tal cual para la determinación de la configuración del vecino. El método continúa posteriormente con la operación 4320. Si el vecino seleccionado está ocupado (sí en la operación 4330), se comprueba en la operación 4350 si este vecino ya se ha codificado o no. Si ya no se ha codificado (no en la operación 4350), el método pasa a la operación 4360 y el vecino todavía no codificado se considerará ocupado en la computación de la configuración de ocupación del vecino. También para un vecino de este tipo, su ocupación se toma tal como es para la determinación de la configuración del vecino. El método continúa posteriormente con la operación 4320. Si el vecino seleccionado ya se ha codificado (sí en la operación 4350), se comprueba en la operación 4370 si al menos uno de los subvolúmenes ocupados del vecino ya codificado comparte una cara con el volumen actual. En términos generales, en la operación 4370 se comprueba si al menos uno de los subvolúmenes ocupados del vecino ya codificado interseca con el volumen actual. Si lo hace (sí en la operación 4370), el método avanza a la operación 4360 y el vecino ya codificado se considerará ocupado en la computación de la configuración de vecino. Por lo tanto, también para dicho vecino, su ocupación se toma como tal para la determinación de la configuración de vecino. De lo contrario (no en la operación 4370), el método avanza a la operación 4340 y el vecino ya codificado se considerará no ocupado en la computación de la configuración de vecino. Es decir, el bit de ocupación de ese vecino se establecerá (deliberadamente/artificialmente) a cero en la determinación de la configuración de vecino. El método continúa posteriormente con la operación 4320. En la operación 4320, se comprueba si hay vecinos del volumen actual que no se han seleccionado. Si es así (sí en la operación 4320), el método vuelve a la operación 4310 para seleccionar el vecino siguiente del volumen actual. Una vez que todos los vecinos se han procesado (no en la operación 4320), la configuración de vecino se calcula en la operación 4380 dependiendo de las respectivas ocupaciones de los vecinos (por ejemplo, bits de ocupación) decididas en la operación 4340 o en la operación 4360. Este computación puede proceder de la misma manera que se ha descrito anteriormente, sin embargo, teniendo en cuenta las ocupaciones de los vecinos del volumen actual como se determina en las operaciones 4340 y 4360. En el mismo, puede decirse que la operación 4340 modifica la ocupación en relación con la determinación directa basándose únicamente en datos de ocupación del vecino respectivo. Asimismo, se puede decir que la configuración de vecino determinada se modifica con respecto a la determinación directa sin considerar los datos de ocupación para los subvolúmenes de los vecinos ya codificados.

La configuración de vecino (modificada) que se determina usando las ocupaciones modificadas de los vecinos puede usarse para seleccionar una o más probabilidades en la operación 4120 del método 4100 y la operación 4210 del método 4200, respectivamente, a menos que un indicador de desactivación en el flujo de bits indique que debería usarse la configuración de vecino original. Esto se describe con más detalle a continuación.

Los métodos 4100, 4200 y 4300 han probado proporcionar ganancias de compresión, en la geometría de las nubes de puntos, por encima del 1 % en relación con la determinación directa de la configuración de vecino sin considerar los subvolúmenes de los vecinos ocupados ya codificados.

Se entiende que los métodos descritos anteriormente en esta memoria no se limitan a vecinos (o subvolúmenes de vecinos) que comparten una cara con el volumen actual. Por ejemplo, los vecinos a un volumen actual pueden ser todos aquellos volúmenes de mismo nivel que comparten una cara o un borde con el volumen actual. Entonces, el criterio en la operación 4370 del método 4300 tendría que sustituirse por una comprobación de si el vecino tiene un subvolumen ocupado que comparte una cara o un borde con el volumen actual. Ejemplos de tal definición de vecino

se ilustran en la Figura 33. Como otro ejemplo, los vecinos a un volumen actual pueden ser todos aquellos volúmenes de mismo nivel que comparten una cara, un borde o un vértice con el volumen actual.

Entonces, el criterio en la operación 4370 del método 4300 tendría que sustituirse por una comprobación de si el vecino tiene un subvolumen ocupado que comparte una cara, un borde o un vértice con el volumen actual. Ejemplos de tal definición de vecino se ilustran en la Figura 34.

En un caso general, los vecinos del volumen actual pueden ser todos aquellos volúmenes de mismo nivel que se intersecan con el volumen actual. Además, independientemente de la definición de los vecinos del volumen actual, el criterio en la operación 4370 del método 4300 puede reemplazarse por una comprobación de si el vecino tiene un subvolumen ocupado que interseca con el volumen actual. En otras palabras, se entiende que los métodos 4100, 4200 y 4300 pueden aplicarse a cualquier árbol de nodos con volúmenes asociados en donde un nodo vecino de un nodo actual se define como un nodo, que tiene la misma profundidad (nivel) que el nodo actual, para el que el volumen asociado tiene una intersección no vacía con el volumen actual asociado con el nodo actual. Por ejemplo, esta intersección puede ser una cara, un borde, un vértice o cualquier conjunto de puntos no vacíos. Un vecino ocupado ya codificado se considerará ocupado en la computación de la configuración de ocupación de vecino si y solo si al menos uno de sus nodos hijos ocupados tiene un volumen asociado que tiene una intersección no vacía con el volumen actual.

Se hace referencia ahora a la Figura 33, que muestra un volumen vecino 4070 que comparte un borde con un volumen actual 4000. En la Figura 33(a), el vecino 4070 ha ocupado los subvolúmenes 4071 y 4072, y ninguno de ellos comparte un borde con el volumen actual 4000. En este caso, el vecino 4070 se considera como no ocupado en la computación de la configuración de ocupación de vecino. En la Figura 33(b), al menos uno de los subvolúmenes 4073 y 4074 del vecino 4070 comparte un borde con el volumen actual 4000. En este caso, el vecino 4070 se considera ocupado en la computación de la configuración de ocupación de vecino.

Se hace referencia ahora a la Figura 34, que muestra un volumen vecino 4080 que comparte un vértice con un volumen actual 4000. En la Figura 34(a), el vecino 4080 ha ocupado los subvolúmenes 4081 y 4082, y ninguno de ellos comparte un vértice con el volumen actual 4000. En este caso, el vecino 4080 se considera como no ocupado en la computación de la configuración de ocupación de vecino. En la Figura 34(b), al menos uno de los subvolúmenes 4083 y 4084 del vecino 4070 comparte un vértice con el volumen actual 4000. En este caso, el vecino 4078 se considera ocupado en la computación de la configuración de ocupación de vecino.

Se ha observado que los métodos 4100, 4200 y 4300 proporcionan más del 1 % de ganancia de compresión, es decir, una reducción de más del 1 % del tamaño del flujo de bits comprimido, en nubes de puntos orientadas a realidad virtual densa. Estas son ganancias interesantes en relación con la simplicidad del método.

Sin embargo, en nubes de puntos dispersas capturadas, por ejemplo, por un Lidar unido a un vehículo en movimiento, estos métodos pueden mostrar poca o ninguna ganancia (o incluso una ligera pérdida en nubes de puntos extremadamente dispersas). Por lo tanto, puede ser ventajoso añadir un indicador en el flujo de bits que señala la activación (valor de indicador 1) o desactivación (valor de indicador 0) de la adaptación de las ocupaciones de vecinos del volumen actual. Desactivación significa que un vecino se considera ocupado/no ocupado en la computación de la configuración de ocupación de vecino independientemente de la ubicación de sus subnodos ocupados.

Patrones de ocupación de árbol de codificación entrópica usando codificación binaria

Algunas de las técnicas descritas anteriormente de uso de información de ocupación de vecinos para codificar la ocupación de árbol se detallan en solicitud de patente europea n.º 18305037.6. Las realizaciones descritas se centran en usar codificación entrópica no binaria del patrón de ocupación, donde se selecciona una distribución de patrón en función de información de ocupación de vecinos. Sin embargo, en algunos casos, el uso de codificadores binarios puede ser más eficiente en términos de implementación de hardware. Además, las actualizaciones sobre la marcha de muchas probabilidades pueden requerir memoria de acceso rápido y computación dentro del corazón del codificador aritmético. Por consiguiente, puede ser ventajoso encontrar métodos y dispositivos para codificar por entropía el patrón de ocupación usando codificadores aritméticos binarios. Sería ventajoso utilizar codificadores binarios si se pudiera hacer sin degradar significativamente el rendimiento de compresión y mientras se protege contra tener un número abrumador de contextos a rastrear.

La utilización de codificadores binarios en lugar de un codificador no binario se refleja en la fórmula de entropía:

$$H(X_1, X_2|Y) = H(X_1|Y) H(X_2|Y, X_1)$$

donde $X = (X_1, X_2)$ es la información no binaria a codificar, e Y es el contexto de la codificación, es decir, la configuración de vecino o distribución de patrones seleccionada. Para convertir la codificación no binaria de X en codificación binaria, la información (X_1, X_2) se parte en la información X_1 y X_2 que puede codificarse por separado sin aumentar la entropía. Para ello, se debe codificar uno de los dos en función del otro, aquí X_2 dependiendo de X_1 . Esto puede extenderse a n bits de información en X . Por ejemplo, para $n=3$:

$$H(X_1, X_2, X_3 | Y) = H(X_1 | Y) H(X_2 | Y, X_1) H(X_3 | Y, X_1, X_2)$$

Se entenderá que como patrón de ocupación, es decir, La secuencia de bits X, se alarga, existiendo más condiciones para codificar bits posteriores en la secuencia. Para un codificador binario (por ejemplo, CABAC) significa un gran aumento en el número de contextos a rastrear y gestionar. Usando un árbol octal como ejemplo, donde el patrón de ocupación es una secuencia de ocho bits $b=b_0...b_7$, la secuencia de bits puede partirse en los ocho bits de información binarios $b_0...b_7$. La codificación puede usar la configuración de vecino N (o NC) para determinar el contexto. Suponiendo que se pueden reducir las configuraciones de vecino a 10 configuraciones de vecino efectivas a través de la agrupación de configuraciones de vecino en clases de invariancia, como se ha descrito anteriormente, entonces N es un número entero que pertenece a $\{0, 1, 2, ..., 9\}$. Para abreviar, las "clases de configuraciones de vecino invariantes" pueden denominarse en esta memoria, a veces, simplemente como las "configuraciones de vecino", aunque se apreciará que este número reducido de configuraciones de vecino puede realizarse en función de la agrupación basada en clases de configuraciones de vecino en función de invariancia.

La Figura 21 ilustra la partición de un patrón o secuencia de ocho bits en ocho bits individuales para la codificación entrópica binaria. Se observará que el primer bit de la secuencia se codifica en función de la configuración de vecino, por lo que hay diez contextos totales disponibles. El siguiente bit de la secuencia se codifica en función de la configuración de vecino y cualquier bit previamente codificado, es decir, bit b_0 . Esto implica 20 contextos disponibles totales: obtenidos como el producto de 10 de N y 2 de b_0 . El bit final, b_7 se codifica por entropía utilizando un contexto seleccionado de 1280 contextos disponibles: obtenido como el producto de 10 de N y 128 del patrón parcial dado por los bits previamente codificados $b_0, ..., b_6$. Es decir, para cada bit el número de contextos (i es decir. Combinaciones posibles de condiciones/dependencias) es el producto del número de configuraciones de vecino definidas (10, en este ejemplo, en función de la agrupación de las 64 configuraciones de vecino en clases), y el número de patrones parciales posibles a partir de la secuencia ordenada de $n-1$ bits previamente codificados (dados por 2^{n-1}).

Como resultado, hay un total de 2550 contextos para mantener en conexión con la codificación binaria del patrón de ocupación. Este es un número excesivamente grande de contextos a rastrear, y la escasez relativa puede causar un rendimiento deficiente debido a la dilución del contexto, particularmente para bits posteriores en la secuencia.

Por consiguiente, en un aspecto, la presente solicitud describe codificadores y decodificadores que determinan si el conjunto de contextos puede reducirse y, si es así, aplican una operación de reducción de contexto para realizar un conjunto más pequeño de contextos disponibles para codificar por entropía al menos parte de un patrón de ocupación usando un codificador binario. En otro aspecto, la presente solicitud describe además codificadores y decodificadores que aplican una o más rondas de reducción de estado usando las mismas operaciones de reducción de contexto con el fin de realizar una selección de contexto eficaz de un número fijo de contextos. En algunas implementaciones, se aplica la reducción de contexto a priori en generar tablas de consulta de contextos y/o condiciones algorítmicas que a continuación se usan por el codificador o decodificador al seleccionar un contexto adecuado. La reducción se basa en una condición comprobable que el codificador y el decodificador evalúan para determinar qué tabla de consulta seleccionar de o cómo indexar/seleccionar de esa tabla de consulta para obtener un contexto seleccionado.

Ahora se hace referencia a la Figura 22, que muestra, en forma de diagrama de flujo, un método 3000 de ejemplo para codificar patrones de ocupación en un codificador en la nube de puntos basado en árbol usando codificación binaria. El método 3000 puede implementarse mediante un codificador o un decodificador. En el caso de un codificador, las operaciones de codificación son de codificación, y en el caso de un decodificador las operaciones de codificación son de decodificación. La codificación y decodificación son codificación y decodificación entrópica basada en el contexto.

El método 3000 de ejemplo es para codificar por entropía un patrón de ocupación, es decir, una secuencia de bits, para un nodo/volumen particular. El patrón de ocupación señala el estado de ocupación de los nodos hijos (subvolúmenes) del nodo/volumen. En el caso de un árbol octal, hay ocho nodos hijos/subvolúmenes. En la operación 3002, se determina la configuración de vecino. La configuración de vecino es el estado de ocupación de uno o más volúmenes vecinos al volumen para el que se va a codificar un patrón de ocupación. Como se ha analizado anteriormente, hay diversas implementaciones posibles para determinar la configuración de vecino. En algunos ejemplos, hay 10 configuraciones de vecino, y las configuraciones de vecino para un volumen actual se identifican en función de la ocupación de los seis volúmenes que comparten una cara con el volumen actual.

En la operación 3004, un índice i a los nodos hijos del volumen actual se establece en 0. A continuación, en la operación 3006 se realiza una evaluación en cuanto a si es posible la reducción de contexto. Diferentes posibles operaciones de reducción de contexto se analizan con más detalle a continuación. La evaluación de si es posible la reducción de contexto puede basarse, por ejemplo, en qué bit en la secuencia de bits se está codificando (por ejemplo, el valor de índice). En algunos casos, la reducción de contexto puede ser posible para bits posteriores en la secuencia pero no para los primeros pocos bits. La evaluación de si la reducción de contexto es posible puede basarse, por ejemplo, en la configuración de vecino ya que ciertas configuraciones de vecino pueden permitir simplificaciones. Se pueden usar factores adicionales para evaluar si la reducción de contexto es posible en algunas implementaciones. Por ejemplo, se puede proporcionar un límite superior B_0 como el número máximo de contextos que un codificador binario puede usar para codificar un bit, y si el número inicial de contextos para codificar un bit es mayor que B_0 , entonces se aplica una reducción de contexto (de lo contrario no lo es) de manera que el número de contextos después

de la reducción es como máximo B_0 . Tal límite B_0 puede definirse en una especificación de codificador y/o decodificador para asegurar que una implementación de software o hardware capaz de tratar con contextos de B_0 siempre será capaz de codificar y/o decodificar una nube de puntos sin generar un desbordamiento en términos del número de contextos. Conocer el límite B_0 de antemano también permite anticipar la complejidad y la huella de memoria inducida por el codificador binario de entropía, facilitando así el diseño del hardware. Los valores típicos para B_0 son de diez a unos pocos cientos.

Si se determina que la reducción de contexto está disponible, entonces en la operación 3008 se aplica una operación de reducción de contexto. La operación de reducción de contexto reduce el número de contextos disponibles en un conjunto de contextos disponibles a un conjunto más pequeño que contiene menos contextos totales. Se recordará que el número de contextos disponibles puede depender, en parte, de la posición de bit en la secuencia, es decir, el índice, puesto que el contexto puede depender de un patrón parcial de bits previamente codificados de la secuencia de bits. En algunas implementaciones, el número de contextos disponibles en el conjunto, antes de la reducción, puede basarse en el número de configuraciones de vecino multiplicado por el número de patrones parciales posibles con los bits previamente codificados. Para un bit en índice i , donde i varía desde 0 a n , el número de patrones parciales puede estar dado por 2^i .

Como se ha indicado anteriormente, en algunas implementaciones las operaciones de reducción de contexto se llevan a cabo antes de la codificación, y los conjuntos de contexto reducidos resultantes son los conjuntos de contexto disponibles para su uso por el codificador y el decodificador durante la operación de codificación. El uso y/o la selección del conjunto de contextos reducidos durante la codificación pueden basarse en la evaluación de una o más condiciones anteriores al uso de esos conjuntos reducidos que corresponden a las condiciones evaluadas en la operación 3006 para determinar que se puede reducir el número de contextos. Por ejemplo, en el caso de una configuración de vecino específica que permite el uso de un conjunto de contexto reducido, el codificador y/o el decodificador pueden determinar primero si se cumple esa condición de configuración de vecino y luego, si es así, usar el conjunto de contexto reducido correspondiente.

En la operación 3010, se determina el contexto para el bit b_i , es decir, seleccionado del conjunto (o conjunto reducido, si existe) de contextos disponibles en función de la configuración de vecino y el patrón parcial de bits previamente codificados en la secuencia de bits. El bit actual es codificado entonces por entropía por un codificador binario usando el contexto seleccionado en la operación 3012.

Si, en la operación 3014, el índice i indica que el bit codificado actualmente es el último bit en la secuencia, es decir, eso i igual $i_{\max}$, entonces el proceso de codificación avanza al siguiente nodo. De lo contrario, el índice i se incrementa en la operación 3016 y el proceso vuelve a la operación 3006.

Se apreciará que en algunas implementaciones, la selección de contexto puede no depender de la configuración de vecino. En algunos casos, puede depender solo del patrón parcial de bits previamente codificados en la secuencia, si los hay.

En la Figura 23 se ilustra un diagrama de bloques simplificado de parte de un codificador 3100 de ejemplo. En esta ilustración se entenderá que el patrón de ocupación 3102 se obtiene a medida que el volumen correspondiente se particiona en nodos hijos y se cicla a través de un búfer FIFO 3104 que mantiene la geometría de la nube de puntos. La codificación del patrón de ocupación 3102 se ilustra como que implica una cascada de codificadores binarios 3106, uno por cada bit del patrón. Entre al menos algunos de los codificadores binarios 3106 hay una operación de reducción de contexto 3108 que funciona para reducir los contextos disponibles a un conjunto más pequeño de contextos disponibles.

Aunque la Figura 23 ilustra una serie de codificadores binarios 3106, en algunas implementaciones solo se usa un codificador binario. En el caso en donde se usa más de un codificador, la codificación puede paralelizarse (parcialmente). Dada la dependencia del contexto de un bit con respecto a los bits anteriores en la secuencia de bits, la codificación del patrón no puede necesariamente paralelizarse completamente, pero puede ser posible mejorar el flujo de trabajo mediante el uso de codificadores binarios en cascada para un patrón para lograr algún grado de paralelización y mejora de velocidad.

Operaciones de reducción de contexto

Los ejemplos anteriores proponen que el proceso de codificación incluye una operación de reducción de contexto con respecto a al menos un bit del patrón de ocupación para reducir el conjunto de contextos disponibles a un conjunto más pequeño de contextos disponibles. En este sentido, la "operación de reducción de contexto" puede entenderse como identificar y consolidar contextos que pueden considerarse duplicados o redundantes en las circunstancias de un bit particular b_i . Como se ha indicado anteriormente, el conjunto de contexto reducido puede determinarse antes de la codificación y puede proporcionarse al codificador y al decodificador, y el codificador y el decodificador determinan si utilizar el conjunto de contexto reducido en función de las mismas condiciones descritas a continuación para reducir el conjunto de contexto.

Reducción de configuración vecina a través de cribado/apantallamiento

Una primera operación de reducción de contexto de ejemplo implica reducir el número de configuraciones de vecino en función de la cribado/apantallamiento. En principio, el estado de ocupación de factores de configuración de vecino de volúmenes vecinos en el proceso de selección de contexto sobre la base de que los volúmenes vecinos ayudan a indicar si es probable o no que el volumen o subvolumen actual esté ocupado. A medida que los bits asociados con los subvolúmenes en el volumen actual se decodifican, entonces también se factorizan en la selección de contexto; sin embargo, la información de subvolúmenes cercanos puede ser más significativa y más informativa que la información de ocupación de un volumen vecino que se ubica en el otro lado de los subvolúmenes del subvolumen actual. En este sentido, los bits previamente decodificados se asocian con subvolúmenes que "criban" o "apantallan" el volumen vecino. Esto puede significar que en tales circunstancias, la ocupación del volumen vecino puede ignorarse ya que la relevancia de su estado de ocupación se incluye por el estado de ocupación de los subvolúmenes entre el subvolumen actual y el volumen vecino, permitiendo de este modo la reducción del número de configuraciones de vecino.

Se hace referencia ahora a la Figura 24, que muestra esquemáticamente una operación de reducción de contexto de ejemplo basada en la cribado de vecinos. El ejemplo implica codificar el patrón de ocupación para un volumen 3200. El patrón de ocupación señala el estado de ocupación de los ocho subvolúmenes dentro del volumen 3200. En este ejemplo, los cuatro subvolúmenes en la mitad superior del volumen 3200 se han codificado, por lo que se conoce su estado de ocupación. El bit del patrón de ocupación que se está codificando se asocia con un quinto subvolumen 3204 que se ubica en la mitad inferior del volumen 3200, por debajo de los cuatro subvolúmenes previamente codificados.

La codificación en este ejemplo incluye determinar el contexto en función de la configuración de vecino. Se muestran las 10 configuraciones de vecino 3202. El volumen 3200 que contiene el quinto subvolumen 3204 que se va a codificar se muestra en gris claro y se indica por el número de referencia 3200. Las configuraciones de vecino 3202 se basan en el estado de ocupación de los volúmenes adyacentes al volumen 3200 y comparten una cara con el mismo. Los volúmenes vecinos incluyen un volumen vecino superior 3206.

En este ejemplo, el número de configuraciones de vecino puede reducirse de 10 a 7 ignorando el volumen vecino superior 3206 en al menos algunas de las configuraciones. Como se muestra en la Figura 24, tres de las cuatro configuraciones en donde se muestra el volumen vecino superior 3206 pueden incluirse en configuraciones equivalentes que no factorizan en el volumen vecino superior 3206, reduciendo de este modo el número de configuraciones de vecino a 7 en total. Puede ser aún ventajoso mantener la configuración que muestra los seis volúmenes vecinos ya que no hay una configuración de vecino de 5 volúmenes existente con la que la configuración de 6 volúmenes se pueda consolidar (habiendo eliminado el uno de 5 elementos) lo que significa que incluso si se elimina el volumen vecino superior, se produce una nueva configuración de vecino de 5 elementos y no se produce ninguna reducción general en los contextos.

El volumen vecino superior 3206 puede eliminarse de las configuraciones de vecino en este ejemplo porque la determinación de contexto para codificar un bit de ocupación asociado con el quinto subvolumen 3204 ya tendrá en cuenta el estado de ocupación de los cuatro subvolúmenes previamente codificados directamente encima de él, que son una mejor indicación de probabilidad y direccionalidad de ocupación para el quinto subvolumen que el estado de ocupación del volumen vecino superior más distante 3206.

El ejemplo anterior en donde el volumen vecino superior 3206 se criba o apantalla por los subvolúmenes previamente codificados cuando se codifica el bit de ocupación correspondiente al quinto subvolumen 3204 es solo un ejemplo. Dependiendo del orden de codificación dentro del volumen 3200 se pueden realizar y explotar varias otras posibles situaciones de cribado/apantallamiento para reducir las configuraciones de vecino disponibles.

Se hace referencia ahora a la Figura 25, que muestra un segundo ejemplo de cribado/apantallamiento. En este ejemplo, el patrón de ocupación para el volumen 3200 se codifica casi por completo. El subvolumen a codificar es el octavo subvolumen y está oculto en la figura en la esquina inferior posterior (no visible). En este caso, se ha codificado el estado de ocupación de los otros siete subvolúmenes. En particular, los subvolúmenes a lo largo de la parte superior (por lo tanto, la reducción en configuraciones de vecino a siete en total) y a lo largo del lado derecho y el lado delantero. Por consiguiente, además de cribar el volumen vecino superior, los subvolúmenes con bits de ocupación previamente codificados apantallan un volumen vecino delantero 3210 y un volumen vecino del lado derecho 3212. Esto puede permitir la reducción de configuraciones de vecino de siete en total a cinco en total, como se ilustra.

Se apreciará que los dos ejemplos anteriores de apantallamiento son ilustrativos y que en algunos casos se pueden consolidar diferentes configuraciones para tener en cuenta diferentes situaciones de apantallamiento. La operación de reducción de contexto basada en apantallamiento/cribado por subvolúmenes previamente codificados es general y no se limita a estos dos ejemplos, aunque se apreciará que no se puede aplicar en el caso del primer subvolumen a codificar ya que requiere que haya al menos un bit de ocupación previamente codificado asociado con un subvolumen cercano para que haya cualquier apantallamiento/cribado.

También se apreciará que el grado de apantallamiento/cribado para justificar la reducción de la configuración de vecino puede ser diferente en diferentes implementaciones. En los dos ejemplos anteriores, los cuatro subvolúmenes que comparten una cara con un volumen vecino se codificaron previamente antes de que ese volumen vecino se

apantalle/crabe y, por lo tanto, se elimine de las configuraciones de vecino. En otros ejemplos, el apantallamiento/crabe parcial puede ser suficiente, por ejemplo, de uno a tres subvolúmenes previamente codificados que comparten una cara.

Reducción de contexto a través de manejo de casos especiales

- 5 Existen ciertos casos en donde la reducción de contexto puede ocurrir sin pérdida de información útil. En el proceso de determinación de contexto de ejemplo descrito anteriormente, el contexto para codificar un bit de ocupación se basa en la configuración de vecino, es decir, el patrón de ocupación de volúmenes vecinos del volumen actual, y sobre el patrón parcial atribuible a la ocupación de subvolúmenes en el volumen actual que se codificaron previamente. Esta última condición da como resultado $2^7 = 128$ contextos a rastrear con respecto al octavo bit en la secuencia de bits del patrón de ocupación. Incluso si las configuraciones de vecino se reducen a cinco en total, esto significa 640 contextos para realizar un seguimiento.

15 El número de contextos es grande en función del hecho de que los bits previamente codificados de la secuencia de bits tienen un orden, y el orden es relevante en la evaluación del contexto. Sin embargo, en algunos casos, el orden puede no contener información útil. Por ejemplo, en el caso en donde la configuración vecina está vacía, es decir, $N_{10} = 0$, se puede suponer que cualquier punto dentro del volumen está escasamente poblado, lo que significa que no tienen una direccionalidad suficientemente fuerte para justificar el seguimiento de contextos separados para diferentes patrones de ocupación en los subvolúmenes hermanos. En el caso de un vecindario vacío, no hay orientación o topología local con respecto a la nube de puntos, lo que significa 2^l condiciones basadas en bits previamente codificados de la secuencia de bits pueden reducirse a j+1 condiciones. Es decir, el contexto para codificar uno de los bits de la secuencia de bits se basa en los bits previamente codificados, pero no en su patrón ordenado, justo en su suma. En otras palabras, la expresión de entropía en este caso especial puede expresarse como:

$$H(b|n) \approx H(b_0|0) H(b_1|0, b_0) H(b_2|0, b_0+b_1) \dots H(b_7|0, b_0+b_1+\dots+b_6)$$

25 En algunas implementaciones, se puede hacer una observación similar con respecto a una configuración de vecino completa. En algunos ejemplos, una configuración de vecino completa carece de direccionalidad, lo que significa que el orden de los bits previamente codificados no necesita tenerse en cuenta en la determinación del contexto. En algunos ejemplos, esta operación de reducción de contexto puede aplicarse a solo algunos de los bits en la secuencia de bits, tal como algunos de los bits posteriores en la secuencia. En algunos casos, la aplicación de esta operación de reducción de contexto a bits posteriores puede estar condicionada a determinar que los bits anteriores asociados con subvolúmenes previamente codificados también estaban todos ocupados.

Reducción de contexto basada en estadística

30 Se puede usar un análisis estadístico para reducir contextos a través de la determinación de cuáles conducen a aproximadamente el mismo comportamiento estadístico y luego combinándolos. Este análisis puede realizarse a priori usando datos de prueba para desarrollar un conjunto de contexto reducido que se proporciona entonces tanto al codificador como al decodificador. En algunos casos, el análisis puede realizarse en una nube de puntos actual usando codificación de dos pasadas para desarrollar un conjunto de contexto reducido personalizado para los datos de nube de puntos específicos.

En algunos de tales casos, la correlación del conjunto de contexto no reducido al conjunto de contexto reducido personalizado puede señalarse al decodificador usando una sintaxis dedicada codificada en el flujo de bits.

40 Se pueden comparar dos contextos mediante un concepto de "distancia". Un primer contexto c tiene una probabilidad p de que un bit b sea igual a cero, y un segundo contexto c' tiene una probabilidad p' de que un bit b' sea igual a cero. La distancia entre c y c' viene dada por:

$$d(c, c') = |p \log_2 p - p' \log_2 p'| + |(1-p) \log_2(1-p) - (1-p') \log_2(1-p')|$$

Utilizando esta medición de similitud (distancia), los contextos pueden agruparse en un proceso, tal como:

1. Comenzar con M_1 contextos y fijar un nivel umbral ϵ
- 45 2. Para un contexto dado, reagrupar en una clase todos los contextos que tienen una distancia desde el contexto dado inferior al nivel umbral ϵ
3. Repetir 2 para todos los contextos no reagrupados hasta que todos se pongan en una clase
4. Etiquetar las M_2 las clases de 1 a M_2 : esto da como resultado una función de reducción de fuerza bruta que correlaciona $\{1, 2, \dots, M_1\} \rightarrow [1, 2, \dots, M_2]$ en donde $M_1 \geq M_2$.
- 50 La función de reducción de fuerza bruta para correlacionar un conjunto de contextos a un conjunto más pequeño de

contextos puede almacenarse en la memoria para ser aplicada por el codificador/decodificador como una operación de reducción de contexto durante la codificación. La correlación puede almacenarse como una tabla de consulta u otra estructura de datos. La función de reducción de fuerza bruta puede aplicarse sólo para bits posteriores en la secuencia de bits (patrón), por ejemplo.

5 Combinaciones y subcombinaciones de operaciones de reducción de contexto

Se han descrito anteriormente tres operaciones de reducción de contexto de ejemplo. Cada una de ellas puede aplicarse individual e independientemente en algunas implementaciones. Dos o más cualesquiera de ellas pueden combinarse en algunas implementaciones. Pueden implementarse operaciones de reducción de contexto adicionales solas o en combinación con una cualquiera o más de las operaciones de reducción de contexto descritas anteriormente.

- 10 La Figura 26 muestra un ejemplo, en forma de diagrama de flujo, de un método 3300 de codificación binaria de patrón de ocupación que implica reducción de contexto combinada. El método 3300 codifica el patrón binario de 8 bits $b_0, b_1, \dots, b_7$, dada una configuración de vecino de 10 elementos N_{10} en $\{0, 1, 2, \dots, 9\}$.

- 15 La primera condición evaluada es si la configuración de vecino está vacía, es decir, $N_{10} = 0$. Si es así, entonces los bits se codifican sin referencia a su orden, como se indica por el número de referencia 3302. De lo contrario, los bits se codifican de manera normal hasta bit b_4 , en cuyo punto el codificador y el decodificador comienzan a aplicar funciones de reducción de contexto de fuerza bruta, BR_i , para reducir el número de contextos al correlacionar el conjunto de contextos definidos por la configuración de vecino y el patrón parcial de bits previamente codificados con un conjunto más pequeño de contextos que tienen resultados estadísticos sustancialmente similares.

- 20 En este ejemplo, los dos últimos bits, b_6 y b_7 se codifican usando configuraciones de vecino reducidas, basadas en apantallamiento/cribado.

- 25 Todas las funciones pueden implementarse como tablas de consulta (LUT) para reducir el tamaño del conjunto de contextos. En una implementación práctica, todas las reducciones se factorizan en las funciones de reducción, es decir, simplemente LUT, que toman los contextos como entrada y proporcionan contextos reducidos como salida. En esta realización de ejemplo, el número total de contextos se ha reducido de 2550 a 576, con el tamaño de salida de cada función de reducción BR_i siendo 70, 106, 110 y 119, respectivamente.

Selección de contexto en sistemas con números fijos de contextos

- 30 Cada una de las operaciones de reducción de contexto descritas anteriormente puede usarse además en un sistema de compresión con un número mínimo estático (fijo) de contextos. En tal diseño, para un símbolo dado en el patrón binario de 8 bits, se aplica una o más operaciones de reducción para determinar el modelo de probabilidad de contexto con el que codificar o decodificar el símbolo.

Impacto en el rendimiento de compresión

- 35 El uso de 10 configuraciones de vecino y codificación no binaria proporciona una ganancia de compresión sobre las implementaciones actuales del modelo de prueba MPEG para codificación de nube de puntos. Sin embargo, el uso propuesto anteriormente de 10 configuraciones de vecino con codificación binaria en cascada usando 2550 contextos da como resultado una mejora incluso mejor en la eficiencia de compresión. Incluso cuando se usa reducción de contexto, tal como usando las tres técnicas detalladas anteriormente, para reducir los contextos a 576 en total, la compresión de codificación binaria es todavía marginalmente mejor que la implementación que usa codificación no binaria, y mucho mejor que el modelo de prueba. Se ha demostrado que esta observación es coherente a través de diferentes datos de nube de puntos de prueba.

- 40 Se hace referencia ahora a la Figura 14, que muestra un diagrama de bloques simplificado de una realización de ejemplo de un codificador 1100. El codificador 1100 incluye un procesador 1102, una memoria 1104 y una aplicación de codificación 1106. La aplicación de codificación 1106 puede incluir un programa informático o aplicación almacenada en la memoria 1104 y que contiene instrucciones que, cuando se ejecutan, hacen que el procesador 1102 realice operaciones tales como las descritas en esta memoria. Por ejemplo, la aplicación de codificación 1106 puede codificar y tiene como salida flujos de bits codificados según los procesos descritos en esta memoria. Se entenderá que la aplicación de codificación 1106 puede almacenarse en un medio no transitorio legible por ordenador, tal como un disco compacto, dispositivo de memoria flash, memoria de acceso aleatorio, disco duro, etc. Cuando se ejecutan las instrucciones, el procesador 1102 lleva a cabo las operaciones y funciones especificadas en las instrucciones para funcionar como un procesador de propósito especial que implementa el (los) proceso(s) descrito(s). Dicho procesador puede denominarse "circuito de procesador" o "circuitaría de procesador" en algunos ejemplos.

- 55 Ahora también se hace referencia a la Figura 15, que muestra un diagrama de bloques simplificado de una realización de ejemplo de un decodificador 1200. El decodificador 1200 incluye un procesador 1202, una memoria 1204 y una aplicación de decodificación 1206. La aplicación de decodificación 1206 puede incluir un programa informático o aplicación almacenada en la memoria 1204 y que contiene instrucciones que, cuando se ejecutan, hacen que el procesador 1202 realice operaciones tales como las descritas en esta memoria. Se entenderá que la aplicación de decodificación 1206 puede almacenarse en un medio legible por ordenador, tal como un disco compacto, un dispositivo

de memoria flash, una memoria de acceso aleatorio, un disco duro, etc. Cuando se ejecutan las instrucciones, el procesador 1202 lleva a cabo las operaciones y funciones especificadas en las instrucciones para funcionar como un procesador de propósito especial que implementa el (los) proceso(s) descrito(s). Dicho procesador puede denominarse "circuito de procesador" o "circuitaría de procesador" en algunos ejemplos.

- 5 Se apreciará que el decodificador y/o el codificador según la presente solicitud pueden implementarse en varios dispositivos informáticos, incluyendo, sin limitación, servidores, ordenadores de propósito general programados adecuadamente, sistemas de visión de máquina y dispositivos móviles. El decodificador o codificador puede implementarse por medio de software que contiene instrucciones para configurar un procesador o procesadores para llevar a cabo las funciones descritas en esta memoria. Las instrucciones de software pueden almacenarse en cualquier
- 10 memoria no transitoria legible por ordenador adecuada, incluyendo CD, RAM, ROM, memoria flash, etc.

- Se entenderá que el decodificador y/o codificador descrito en esta memoria y el módulo, rutina, proceso, subproceso u otro componente de software que implemente el método/proceso descrito para configurar el codificador o decodificador pueden realizarse usando técnicas y lenguajes de programación informática estándar. La presente solicitud no se limita a procesadores particulares, lenguajes informáticos, convenciones de programación informática, estructuras de datos, otros detalles de implementación de este tipo. Los expertos en la técnica reconocerán que los
- 15 procesos descritos pueden implementarse como una parte de código ejecutable por ordenador almacenado en memoria volátil o no volátil, como parte de un chip integrado específico de la aplicación (ASIC), etc.

La presente solicitud también proporciona una señal legible por ordenador que codifica los datos producidos a través de la aplicación de un proceso de codificación según la presente solicitud.

- 20 Se pueden realizar ciertas adaptaciones y modificaciones de las realizaciones descritas. Por lo tanto, las realizaciones anteriormente analizadas se consideran ilustrativas y no restrictivas.

REIVINDICACIONES

1. Un método implementado por ordenador para codificar una nube de puntos para generar un flujo de bits de datos comprimidos de nube de puntos, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolumenes y que contiene los puntos de la nube de puntos, comprendiendo el método:
5 para cada uno de una pluralidad de nodos actuales, cada nodo actual asociado a dicho subvolumen se parte en subvolumenes adicionales, y cada subvolumen adicional corresponde a un nodo hijo del nodo actual:

10 determinar un patrón de ocupación para el nodo actual en función de los estados de ocupación de los nodos hijos indicados por un indicador de ocupación para cada nodo hijo, en donde el patrón de ocupación se especifica mediante un número entero que representa una secuencia de dichos indicadores de ocupación asociados con los nodos hijos en un orden de exploración predefinido;

15 seleccionar una o más probabilidades asociadas con codificadores entrópicos respectivos para codificar por entropía el patrón de ocupación, en donde la selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación para nodos hijos de al menos uno de la pluralidad de nodos vecinos; y

20 codificar por entropía el patrón de ocupación en función de la una o más probabilidades seleccionadas, usando el uno o más codificadores entrópicos asociados, para producir datos codificados para el flujo de bits,

en donde la selección de una o más probabilidades se basa en una configuración de vecino que se determina sobre la base de un estado de ocupación de cada uno de los nodos vecinos del nodo actual, y

en donde los nodos vecinos de los nodos actuales son aquellos nodos que están a la misma profundidad en la estructura de árbol que el nodo actual y cuyos subvolumenes asociados intersecan el subvolumen del nodo actual.
2. Un método implementado por ordenador para decodificar un flujo de bits de datos comprimidos de nube de puntos para producir una nube de puntos reconstruida, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolumenes y que contiene los puntos de la nube de puntos, comprendiendo el método:
25 para cada uno de una pluralidad de nodos actuales, cada nodo actual asociado a dicho subvolumen se parte en subvolumenes adicionales, y cada subvolumen adicional corresponde a un nodo hijo del nodo actual:

30 seleccionar una o más probabilidades asociadas con codificadores entrópicos respectivos para decodificar por entropía el patrón de ocupación para el nodo actual, en donde el patrón de ocupación se basa en estados de ocupación de los nodos hijos indicados por un indicador de ocupación para cada nodo hijo, en donde el patrón de ocupación se especifica mediante un número entero que representa una secuencia de dicho indicadores de ocupación asociados con los nodos hijos en un orden de exploración predefinido, y en donde la selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación de nodos hijos de al menos uno de la pluralidad de nodos vecinos;
35 y seleccionar una o más probabilidades asociadas con los nodos hijos en un orden de exploración predefinido, y en donde la selección se basa en datos de ocupación para una pluralidad de nodos vecinos del nodo actual y en datos de ocupación de nodos hijos de al menos uno de la pluralidad de nodos vecinos;
y

40 decodificar por entropía el flujo de bits sobre la base de la una o más probabilidades seleccionadas, usando el uno o más codificadores entrópicos asociados, para producir un patrón de ocupación reconstruido para la ocupación de señalización de nodo actual de los nodos hijos,

en donde la selección de una o más probabilidades se basa en una configuración de vecino que se determina sobre la base de un estado de ocupación de cada uno de los nodos vecinos del nodo actual, y

45 en donde los nodos vecinos de los nodos actuales son aquellos nodos que están a la misma profundidad en la estructura de árbol que el nodo actual y cuyos subvolumenes asociados intersecan el subvolumen del nodo actual.
3. El método de la reivindicación 1 o 2, en donde un nodo vecino del nodo actual se considera ocupado con el propósito de determinar la configuración de vecino si sus datos de ocupación indican que está ocupado y los datos de ocupación para sus nodos hijos indican que al menos uno de sus nodos hijos ocupados es vecino al nodo actual.
50
4. El método de una cualquiera de las reivindicaciones 1 a 3, en donde un nodo vecino del nodo actual se considera ocupado con el propósito de determinar la configuración de vecino si aún no se ha codificado.
5. El método de una cualquiera de la reivindicación 3 o la reivindicación 4 cuando depende de la reivindicación 3,

en donde los nodos hijos vecinos al nodo actual son aquellos nodos que están a una profundidad más baja en uno en la estructura de árbol que el nodo actual y cuyos subvolúmenes asociados intersecan el subvolumen del nodo actual.

6. El método según una cualquiera de las reivindicaciones anteriores, en donde los datos de ocupación para la pluralidad de nodos vecinos comprenden estados de ocupación para cada uno de la pluralidad de nodos vecinos.

5 7. El método de una cualquiera de las reivindicaciones anteriores, en donde la estructura de árbol representa un árbol octal.

8. El método según una cualquiera de la reivindicación 2 o las reivindicaciones 3 a 7 cuando dependen de la reivindicación 2, que comprende además decodificar un indicador del flujo de bits, indicando el indicador que la una o más probabilidades asociadas con los codificadores entrópicos respectivos para decodificar por entropía el patrón de ocupación deben seleccionarse en función de los datos de ocupación para la pluralidad de nodos vecinos del nodo actual y en los datos de ocupación de los nodos hijos de al menos uno de la pluralidad de nodos vecinos.

9. Un codificador para codificar una nube de puntos para generar un flujo de bits de datos comprimidos de nube de puntos, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolúmenes y que contiene los puntos de la nube de puntos, comprendiendo el codificador:

un procesador;

una memoria; y

una aplicación de codificación que contiene instrucciones ejecutables por el procesador que, cuando se ejecutan, hacen que el procesador realice el método reivindicado en una cualquiera de la reivindicación 1 o las reivindicaciones 3 a 7 cuando dependen de la reivindicación 1.

10. Un decodificador para decodificar un flujo de bits de datos comprimidos de nube de puntos para producir una nube de puntos reconstruida, definiéndose la nube de puntos en una estructura de árbol que tiene una pluralidad de nodos que tienen relaciones padre-hijo y que representan la geometría de un espacio volumétrico partido recursivamente en subvolúmenes y que contiene los puntos de la nube de puntos, comprendiendo el decodificador:

un procesador;

una memoria; y

una aplicación de decodificación que contiene instrucciones ejecutables por el procesador que, cuando se ejecutan, hacen que el procesador realice el método reivindicado en una cualquiera de las reivindicaciones 2 o 3 a 8 cuando depende de la reivindicación 2.

11. Un medio no transitorio legible por procesador que almacena instrucciones ejecutables por procesador que, cuando se ejecutan por un procesador, hacen que el procesador realice el método reivindicado en una cualquiera de las reivindicaciones 1 a 8.

12. Una señal legible por ordenador que contiene instrucciones de programa que, cuando se ejecutan por un ordenador, hacen que el ordenador realice el método descrito reivindicado en una cualquiera de las reivindicaciones 1 a 8.

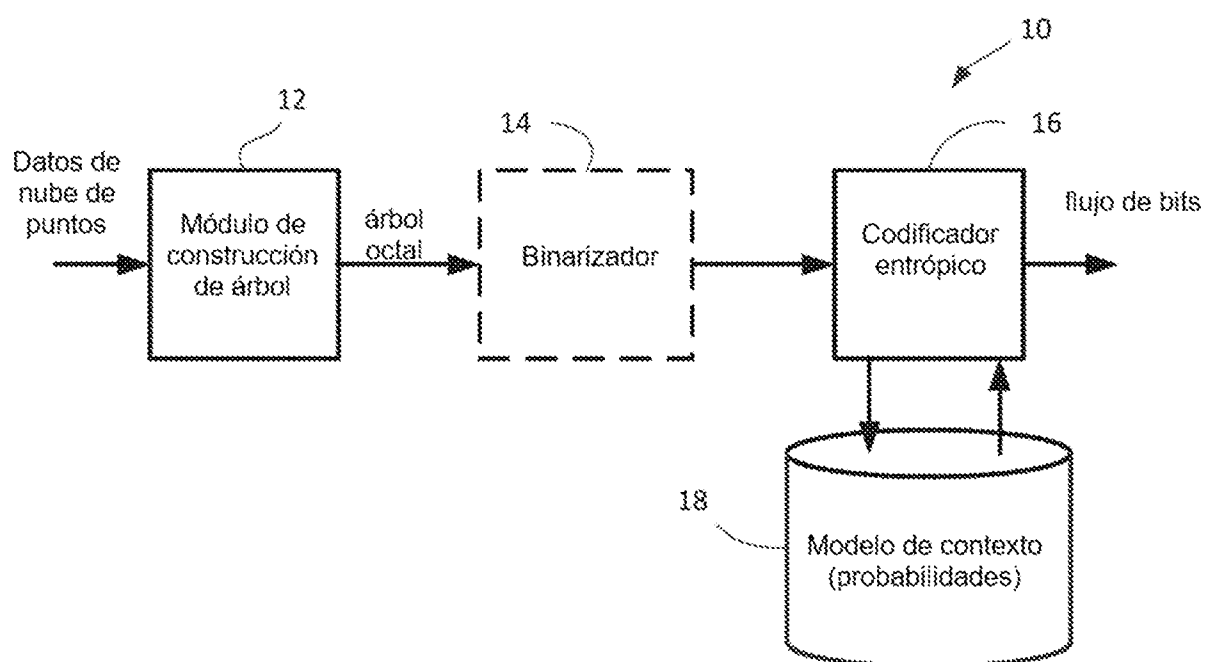


FIG. 1

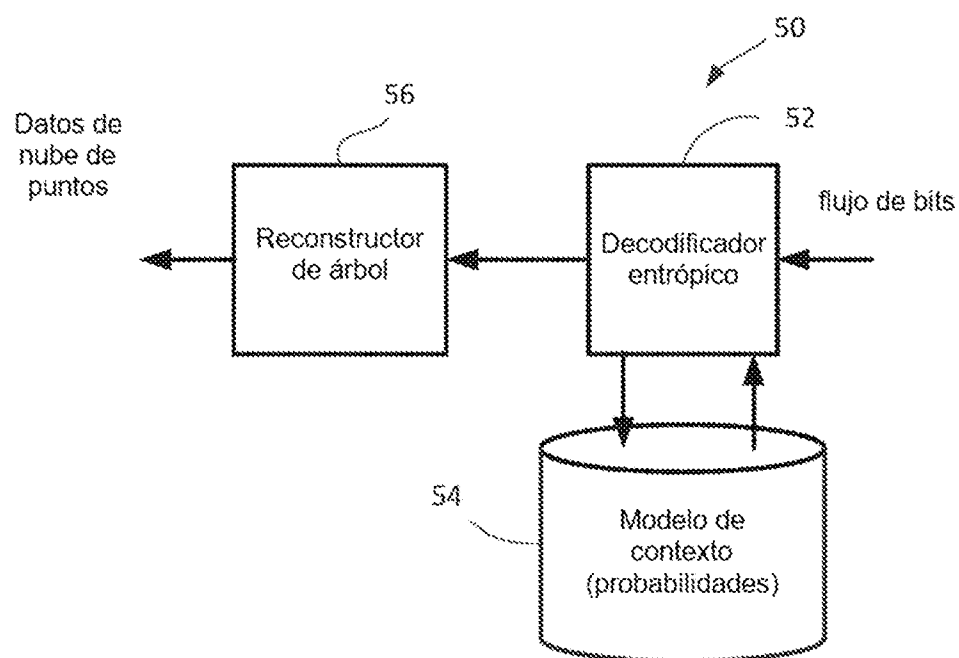


FIG. 2

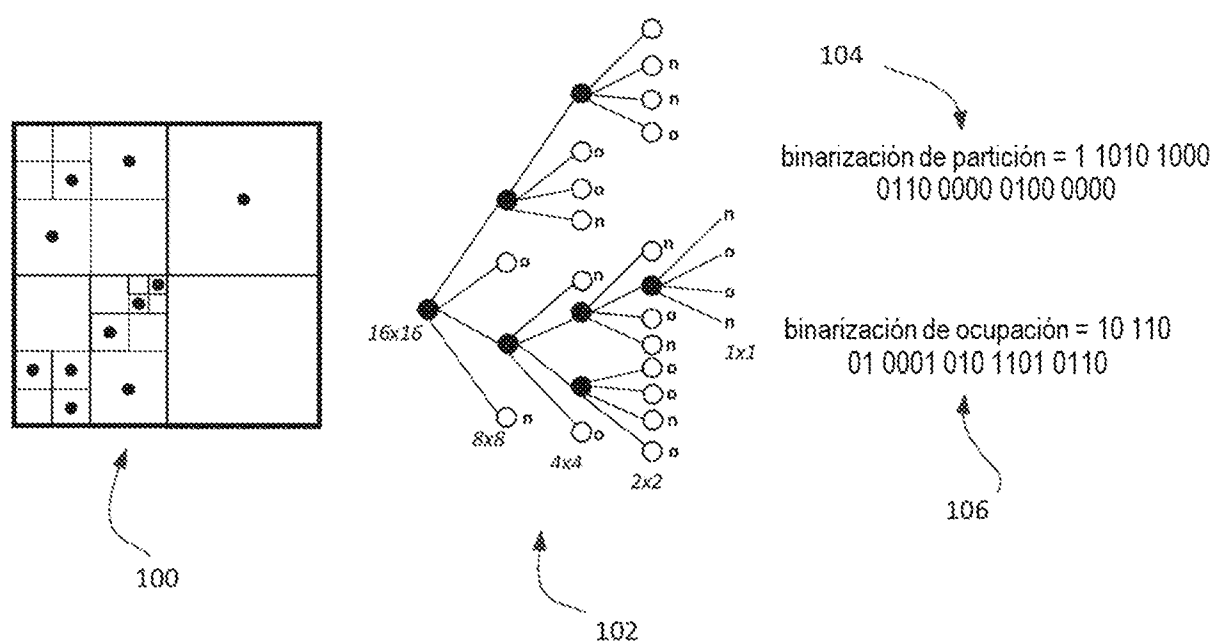


FIG. 3

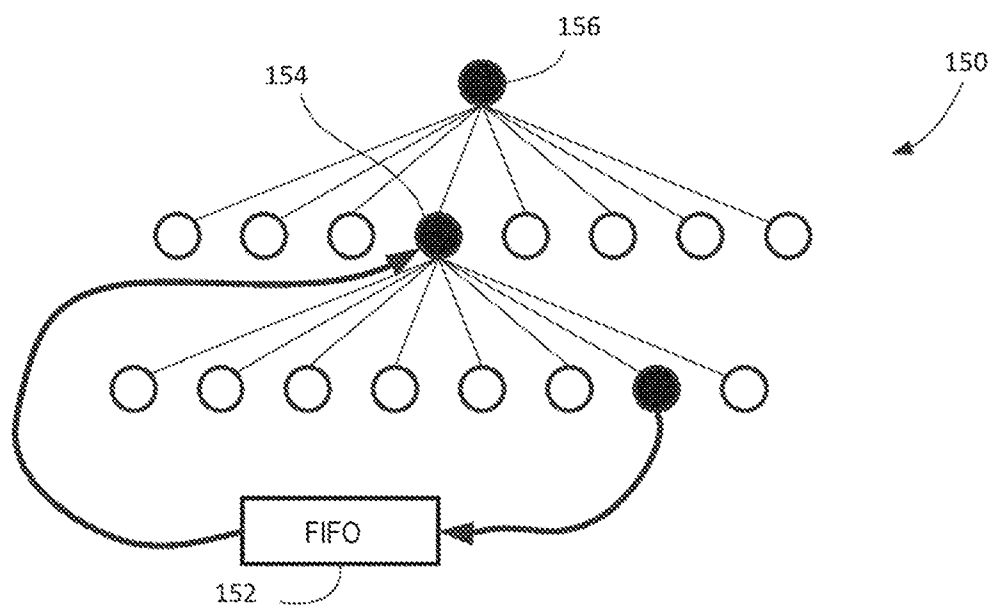


FIG. 4

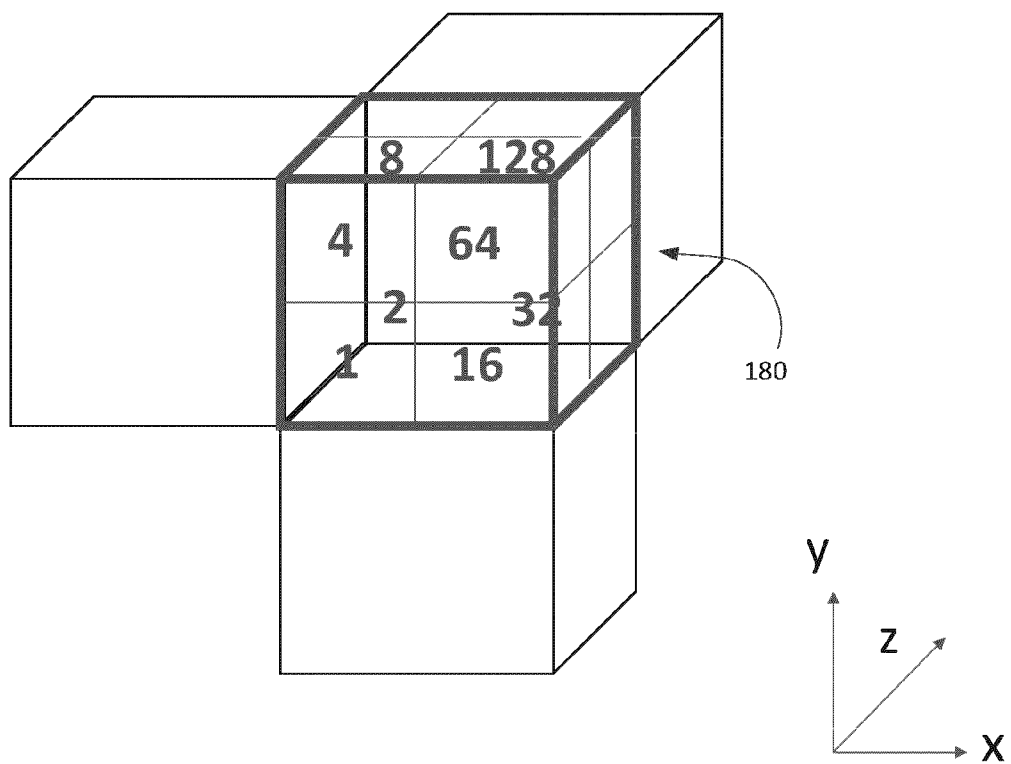


FIG. 5

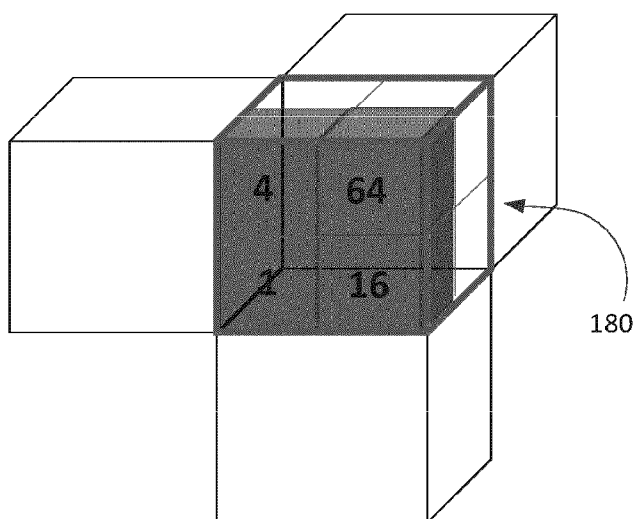
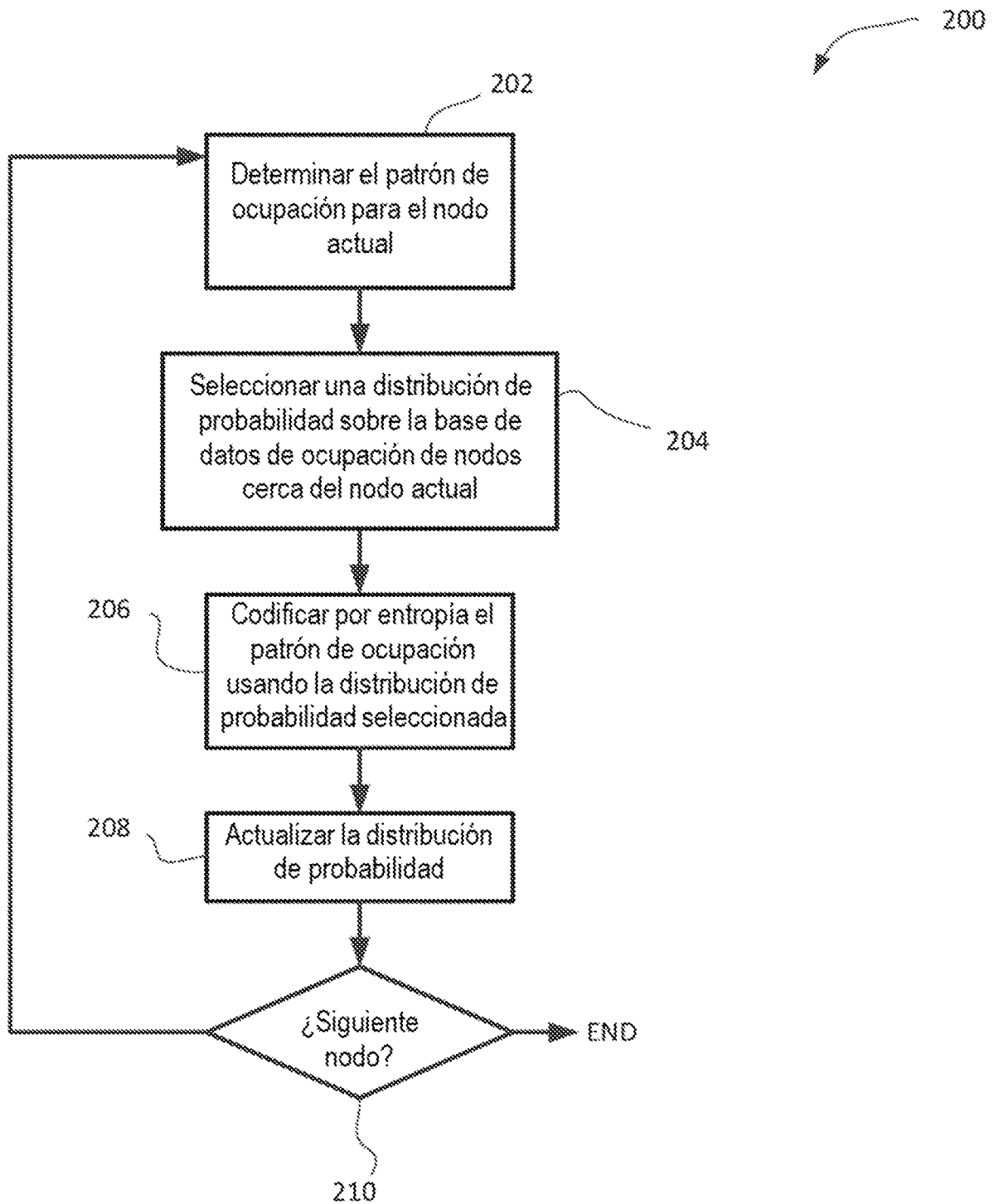


FIG. 6

**FIG. 7**

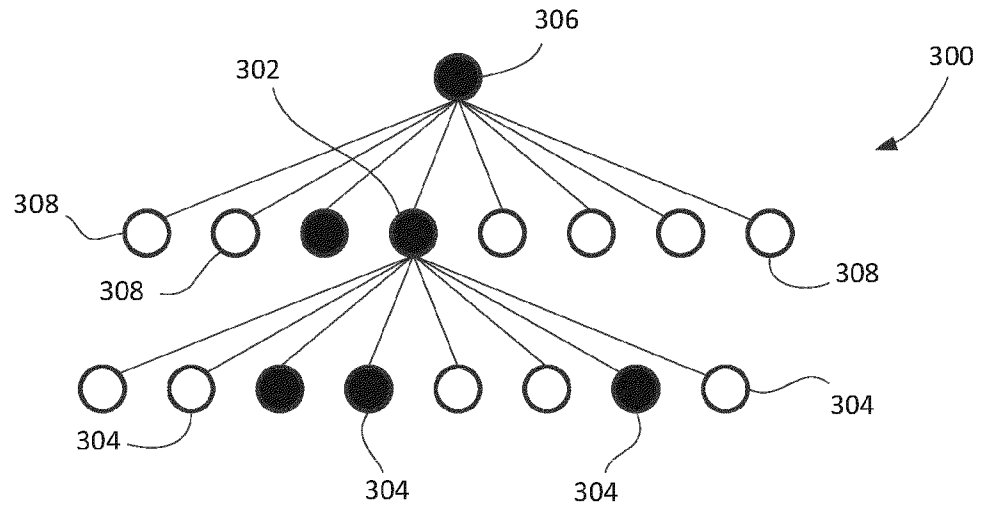


FIG. 8

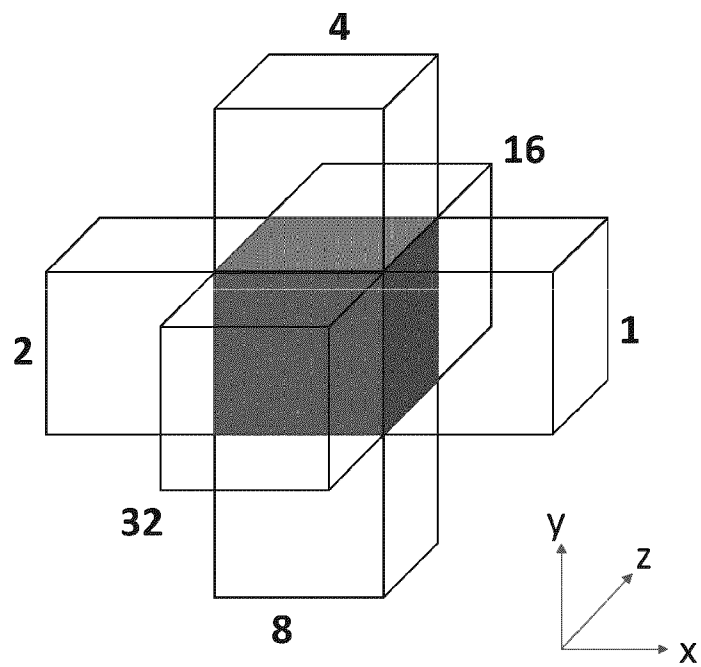


FIG. 9

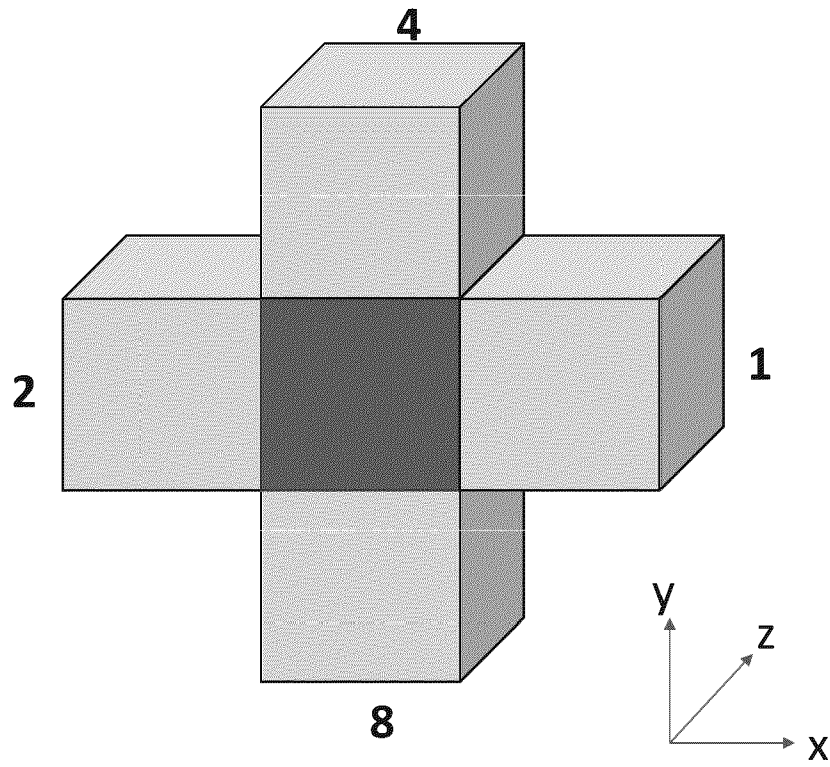
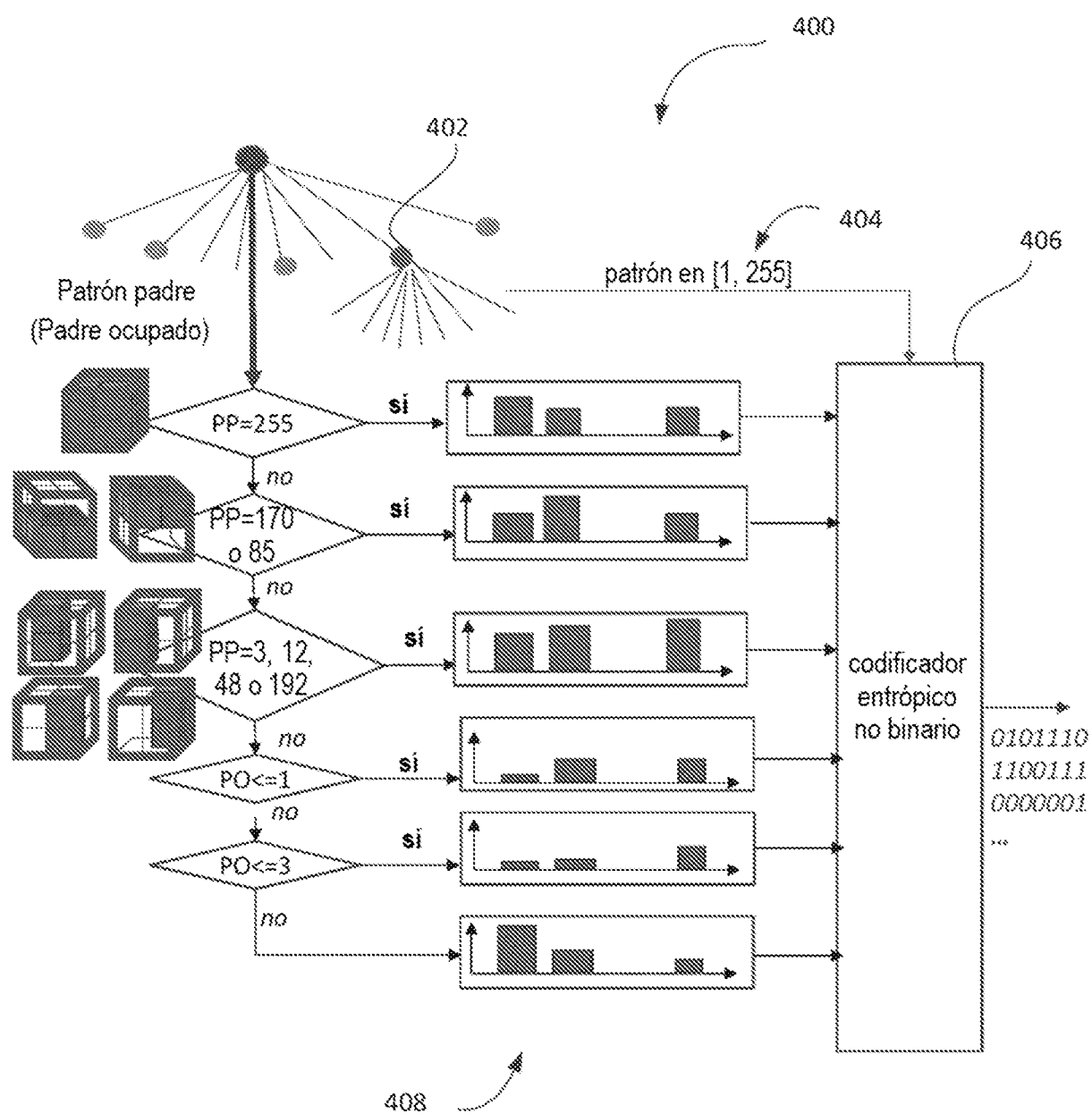
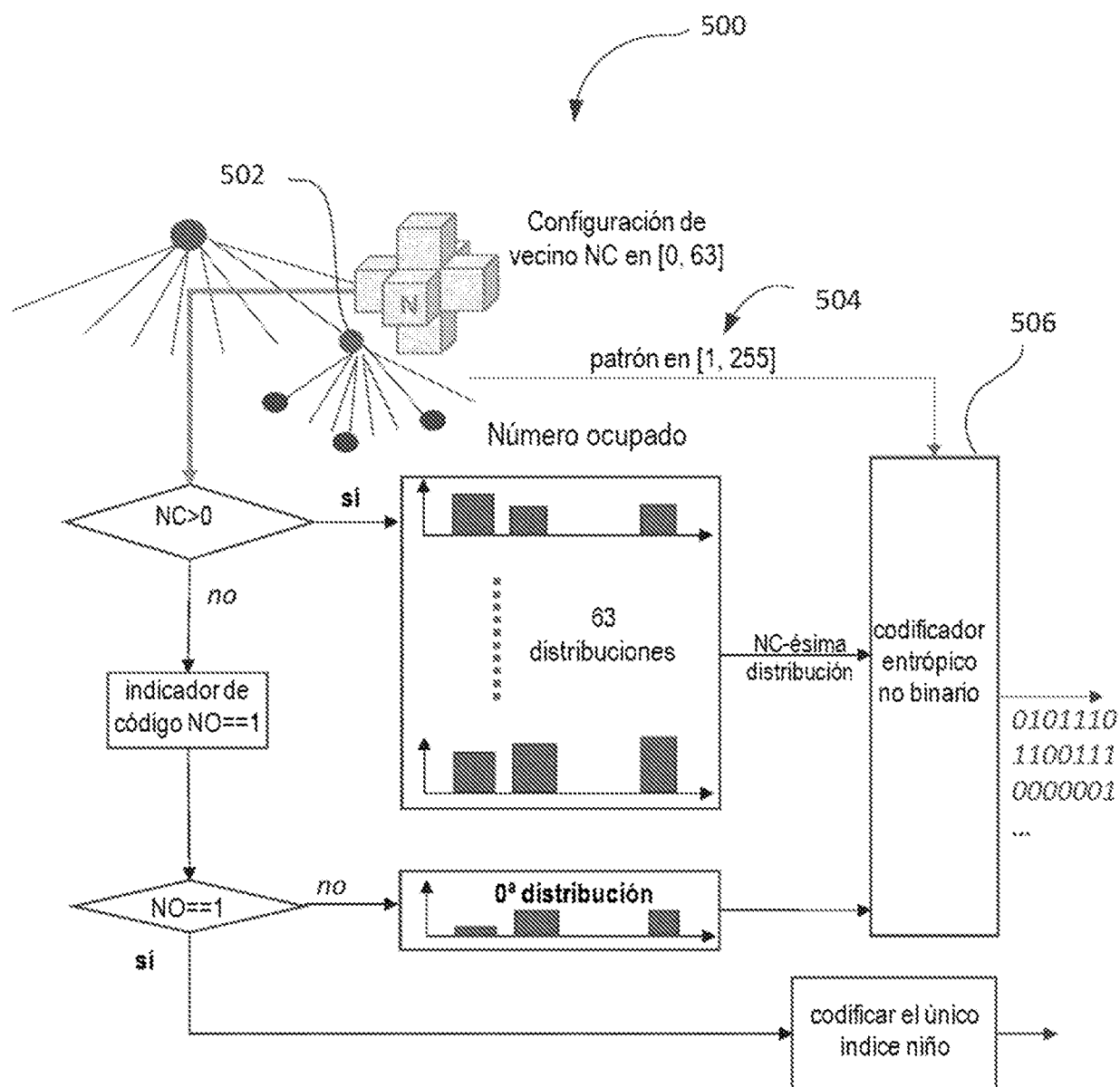
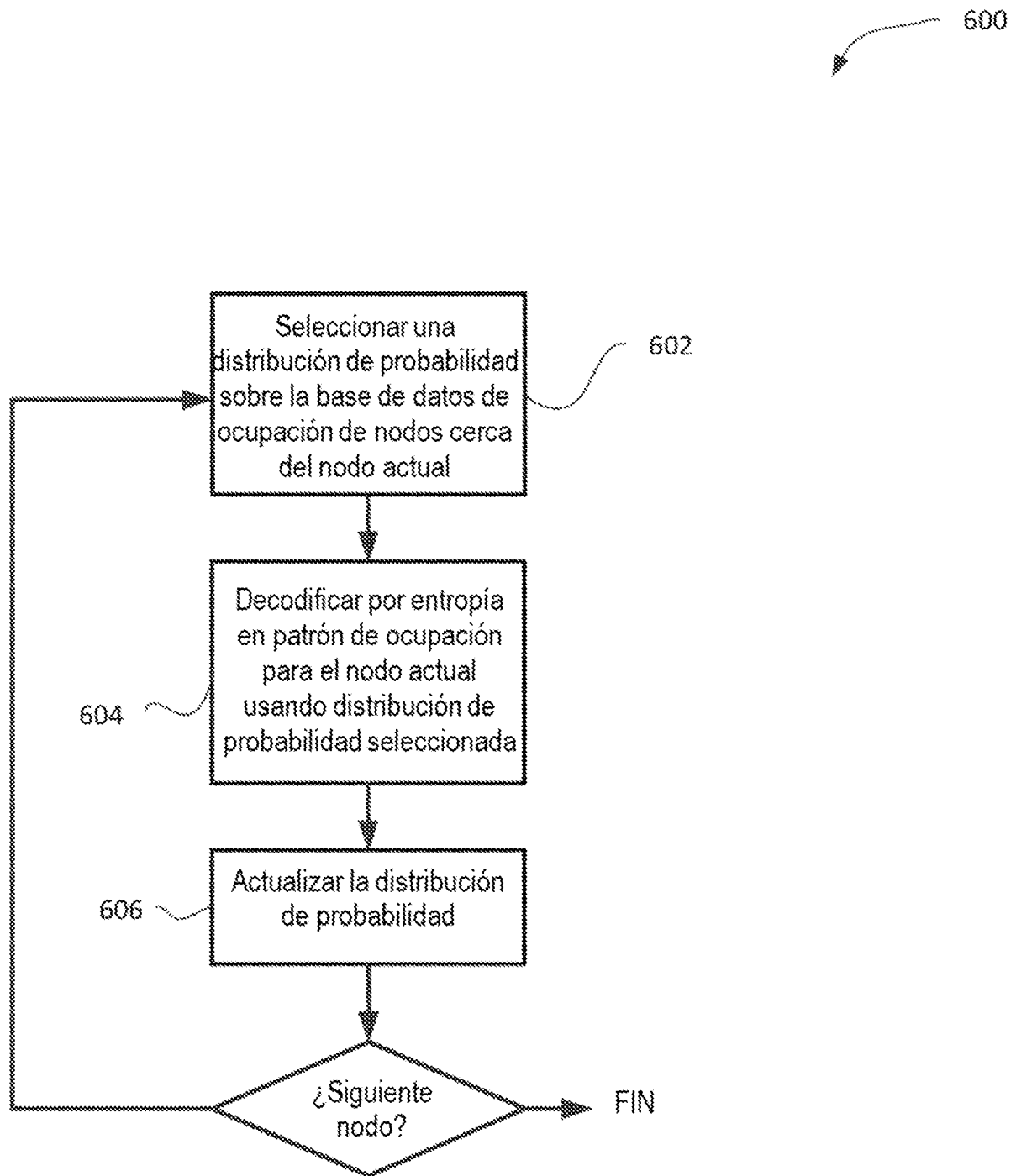


FIG. 10

**FIG. 11**

**FIG. 12**

**FIG. 13**

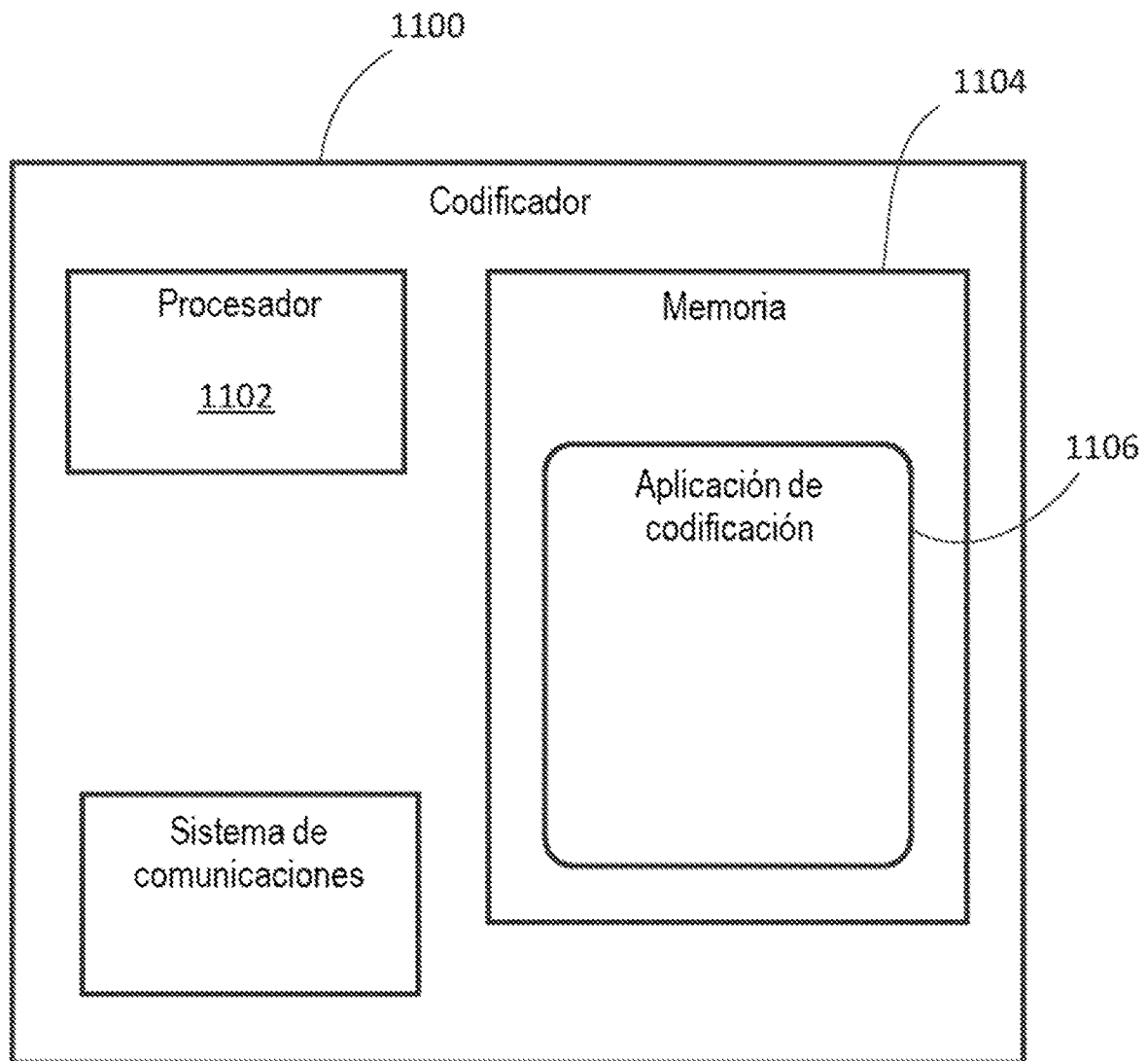


FIG. 14

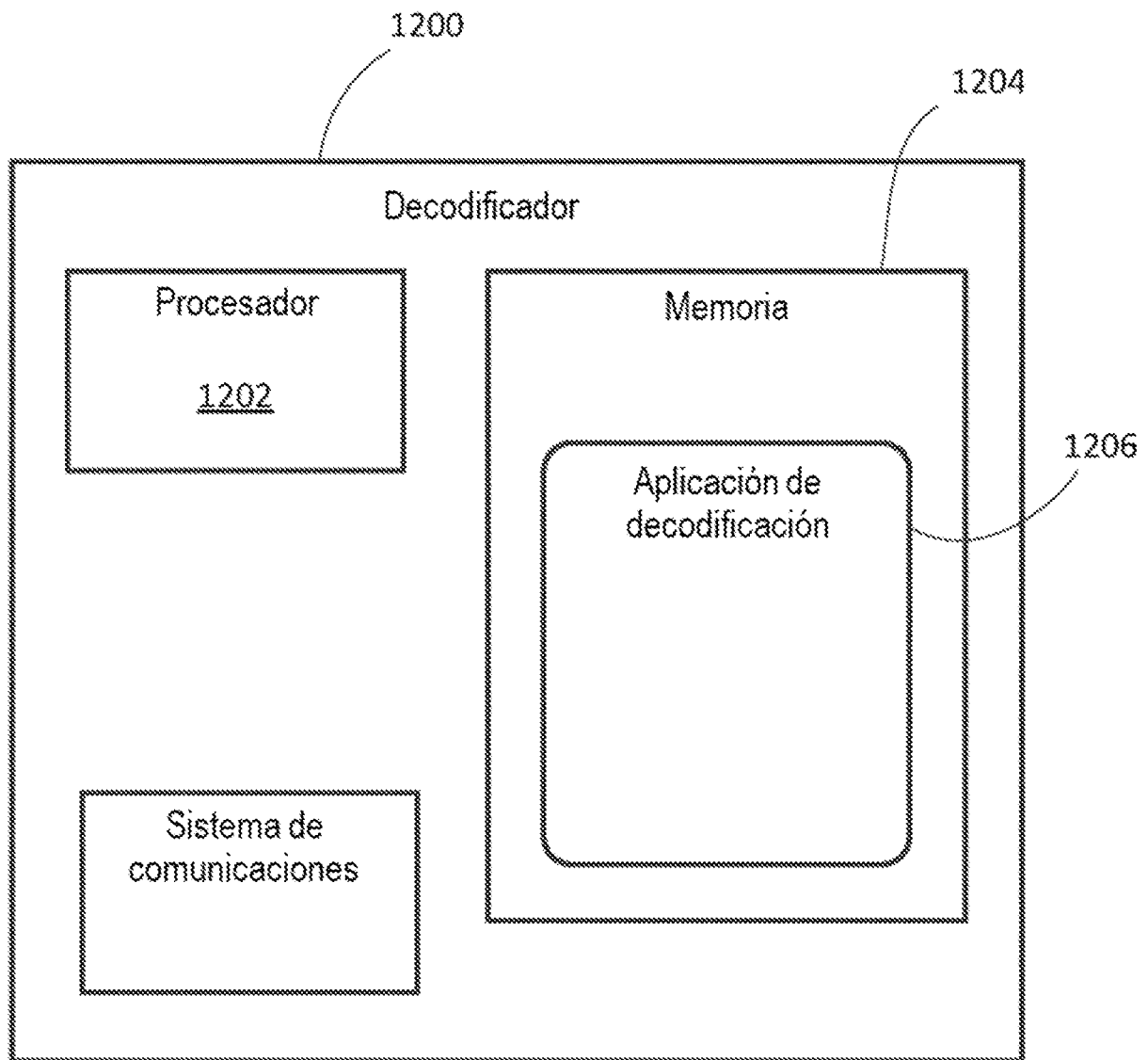


FIG. 15

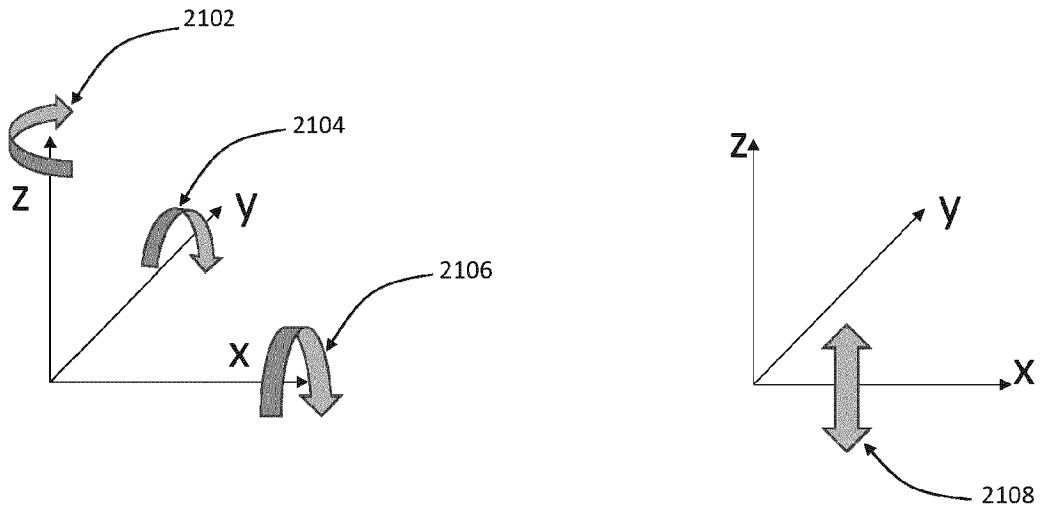


FIG. 16

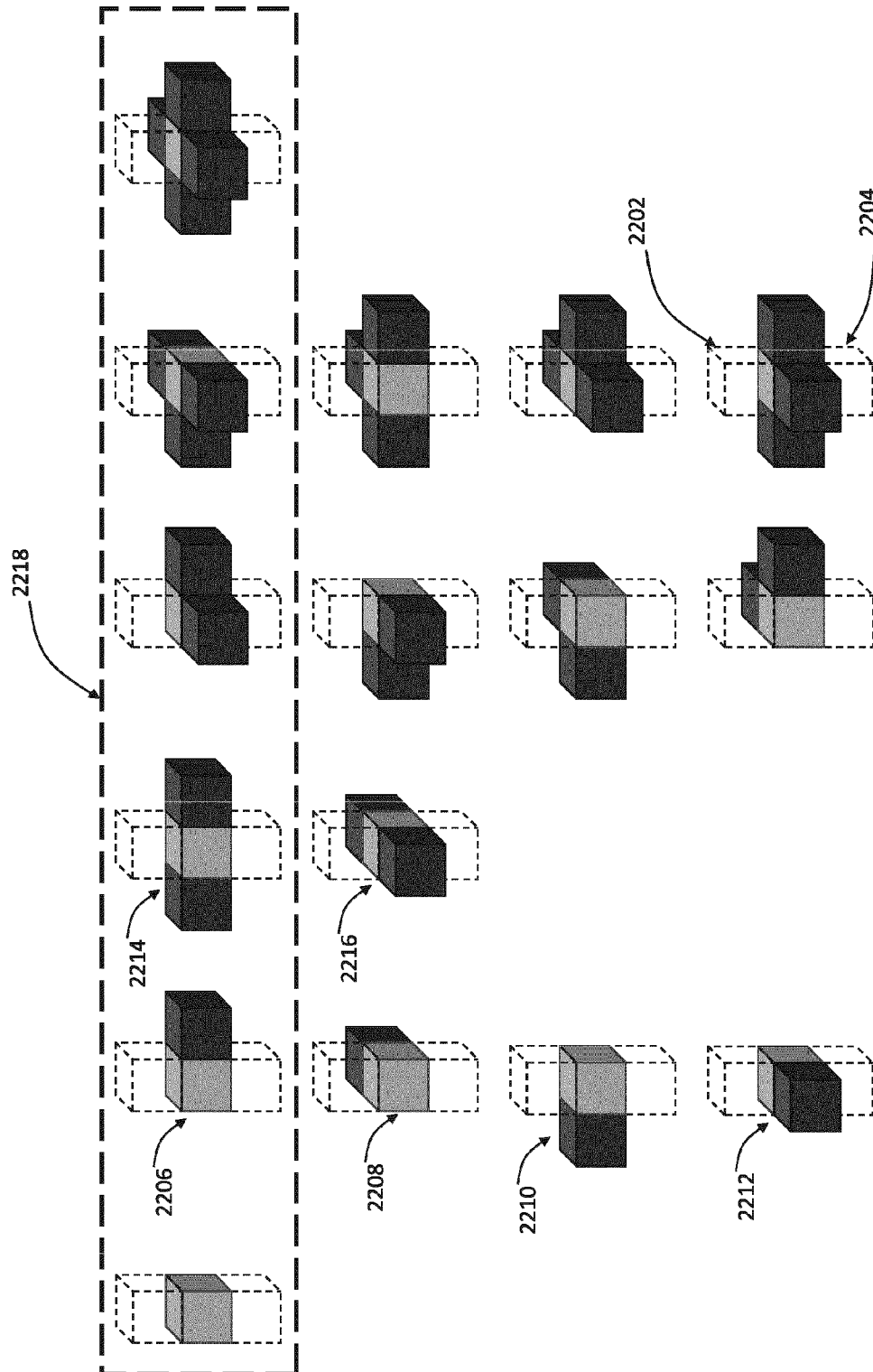


FIG. 17

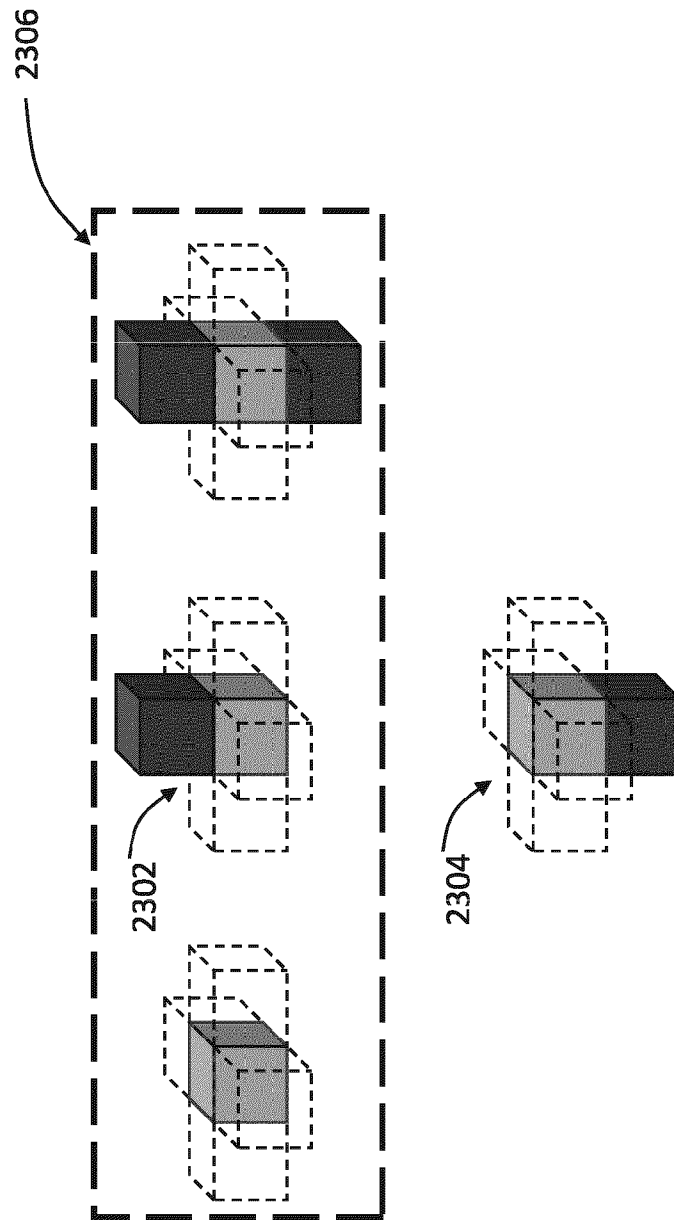


FIG. 18

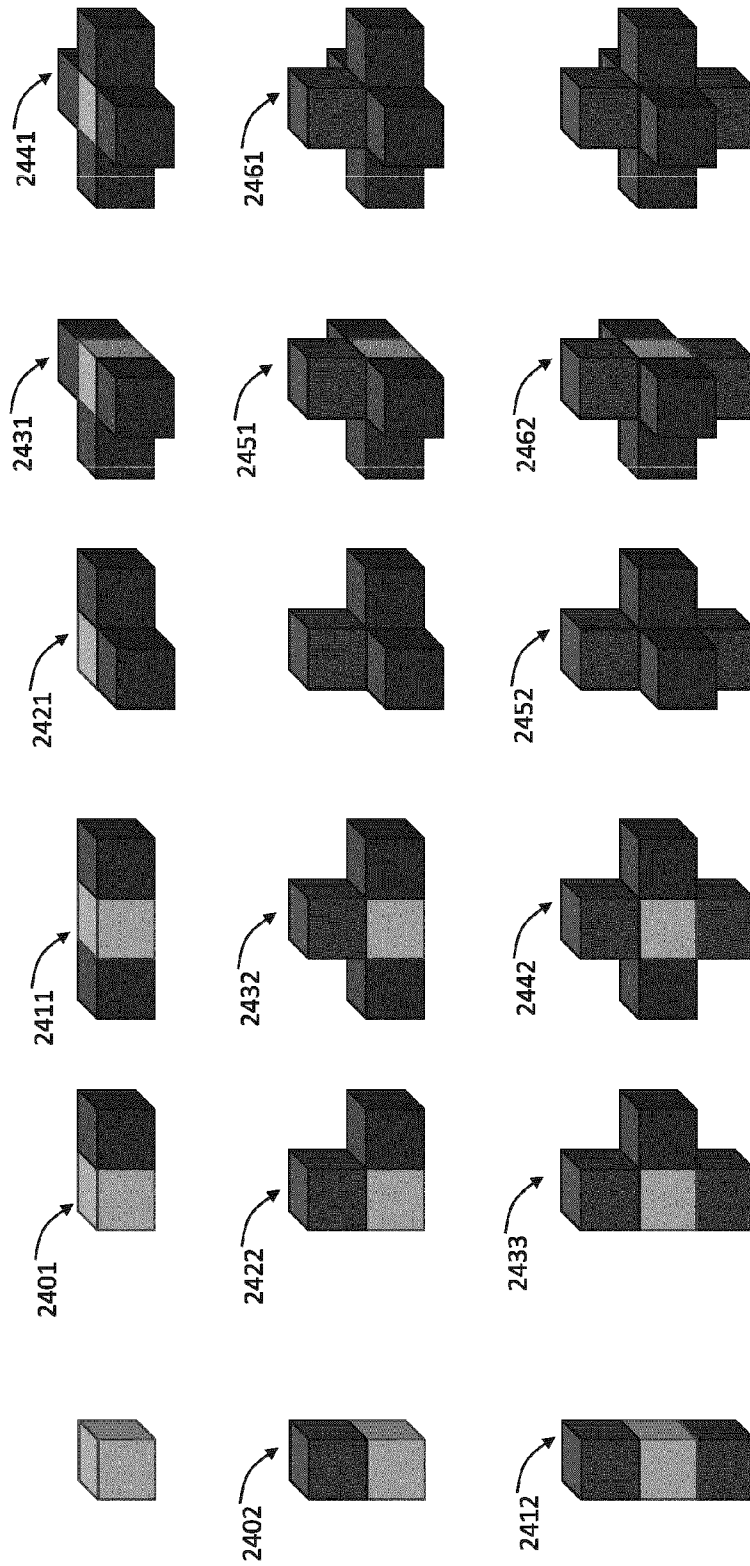


FIG. 19

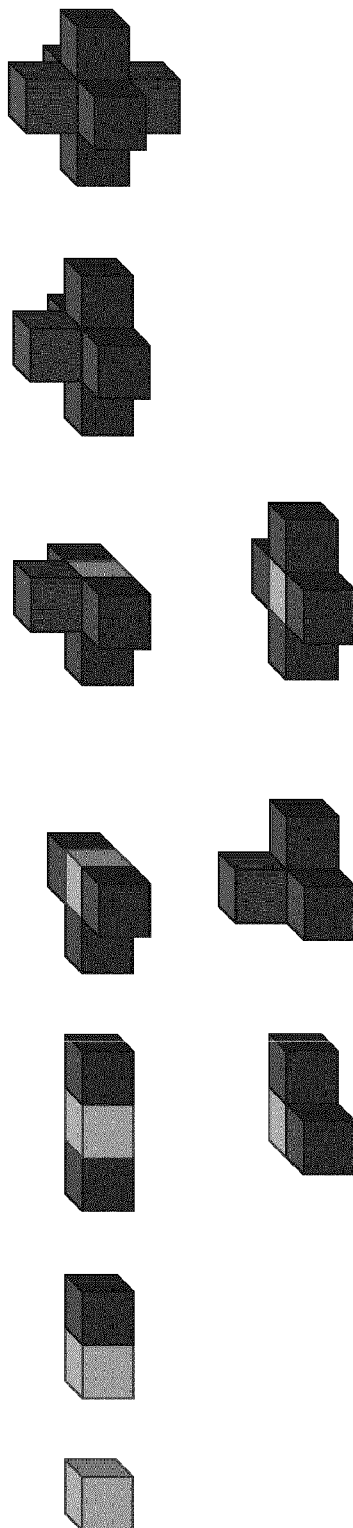


FIG. 20

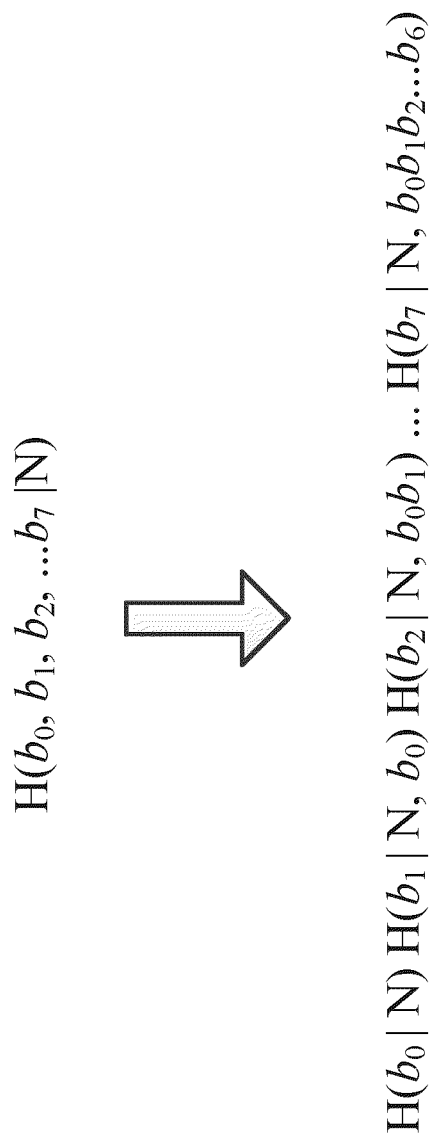
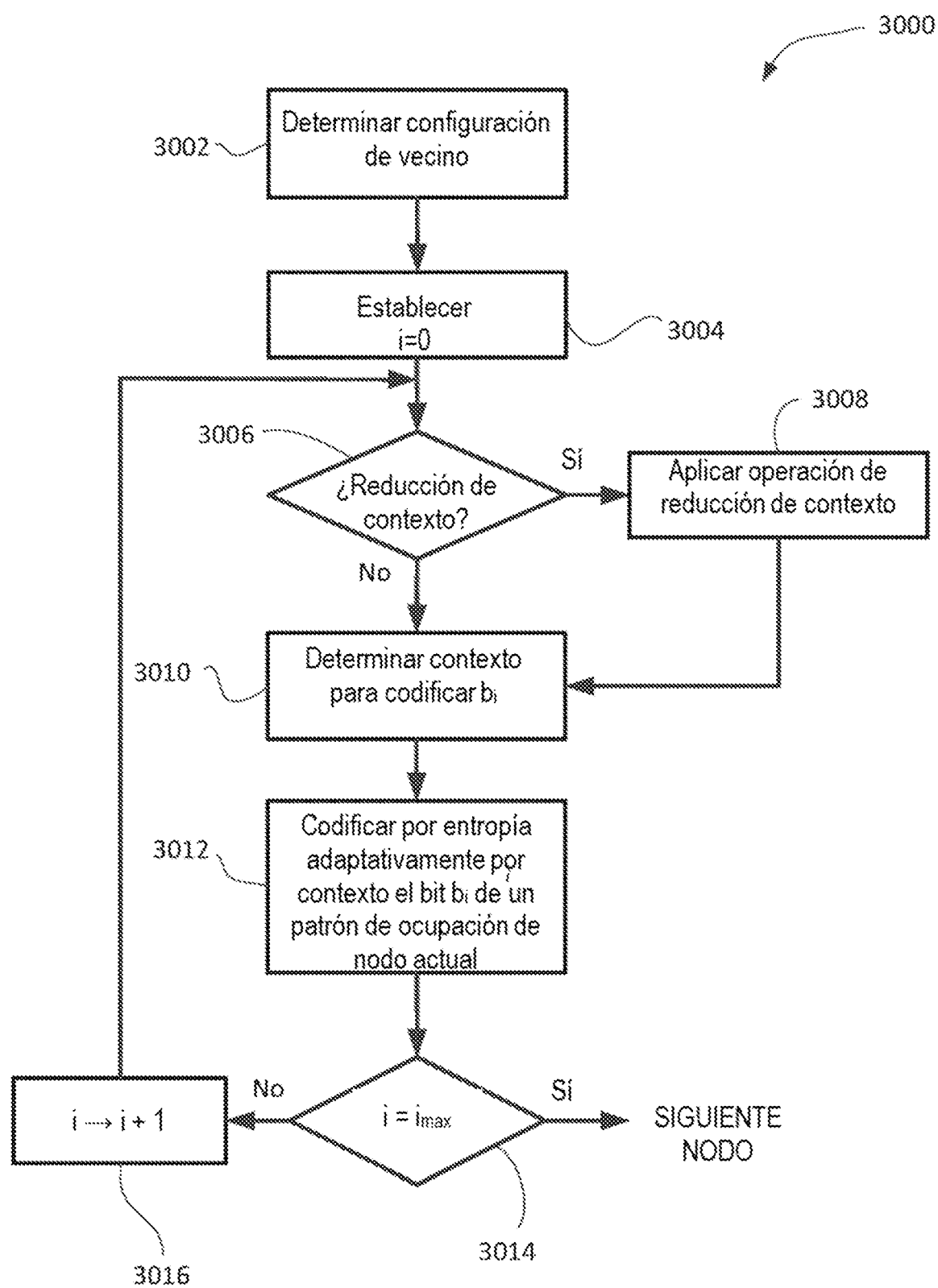
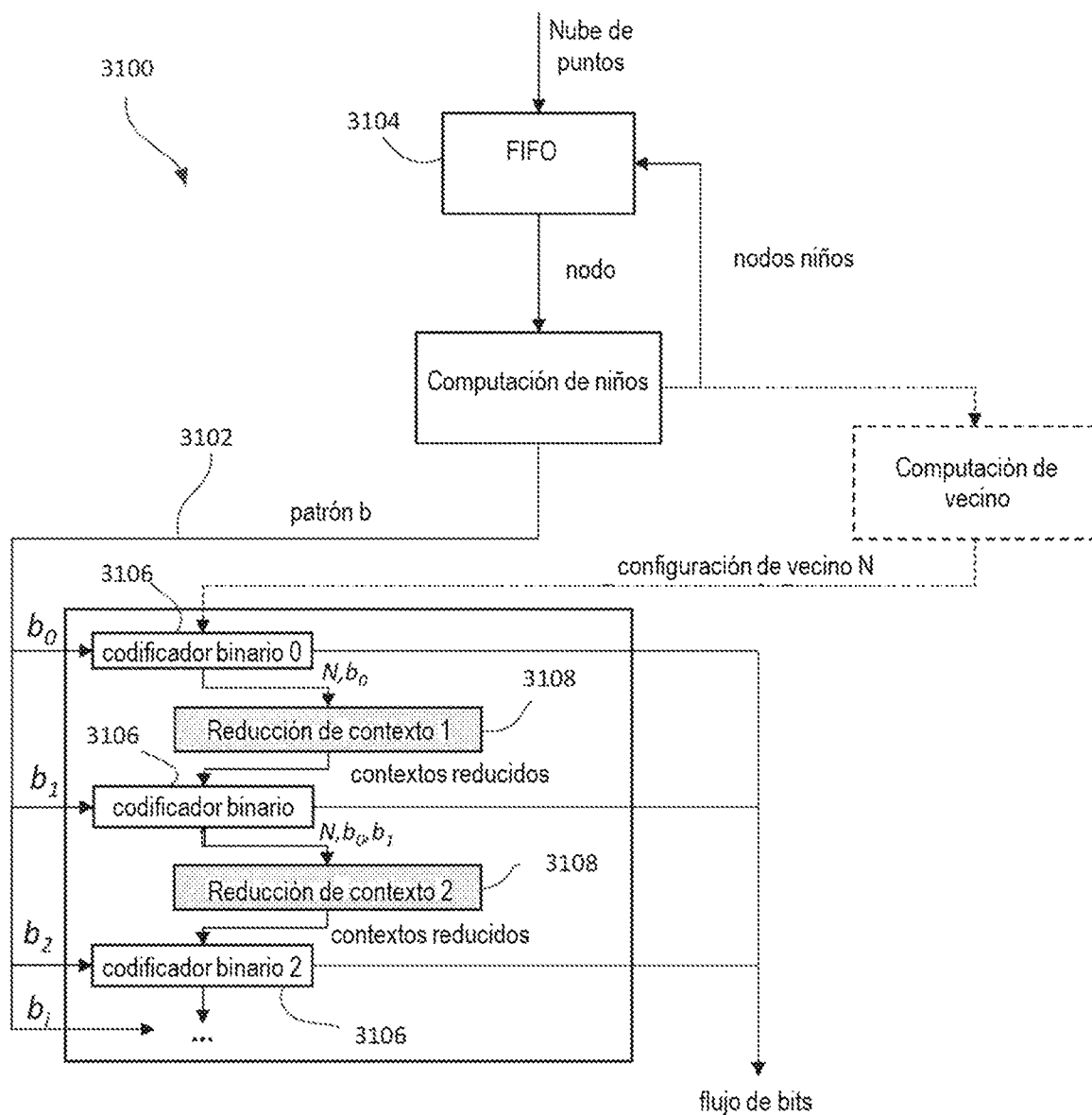


FIG. 21

**FIG. 22**

**FIG. 23**

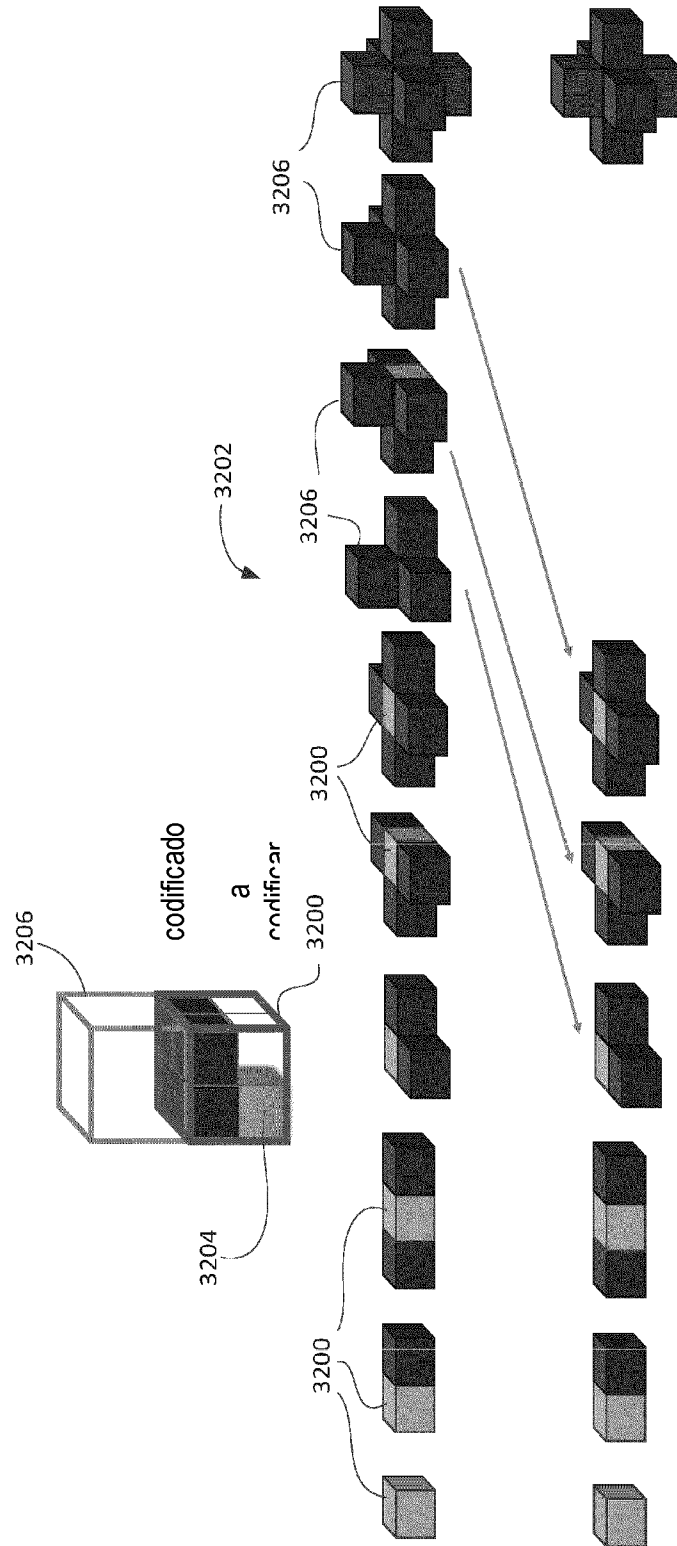


FIG. 24

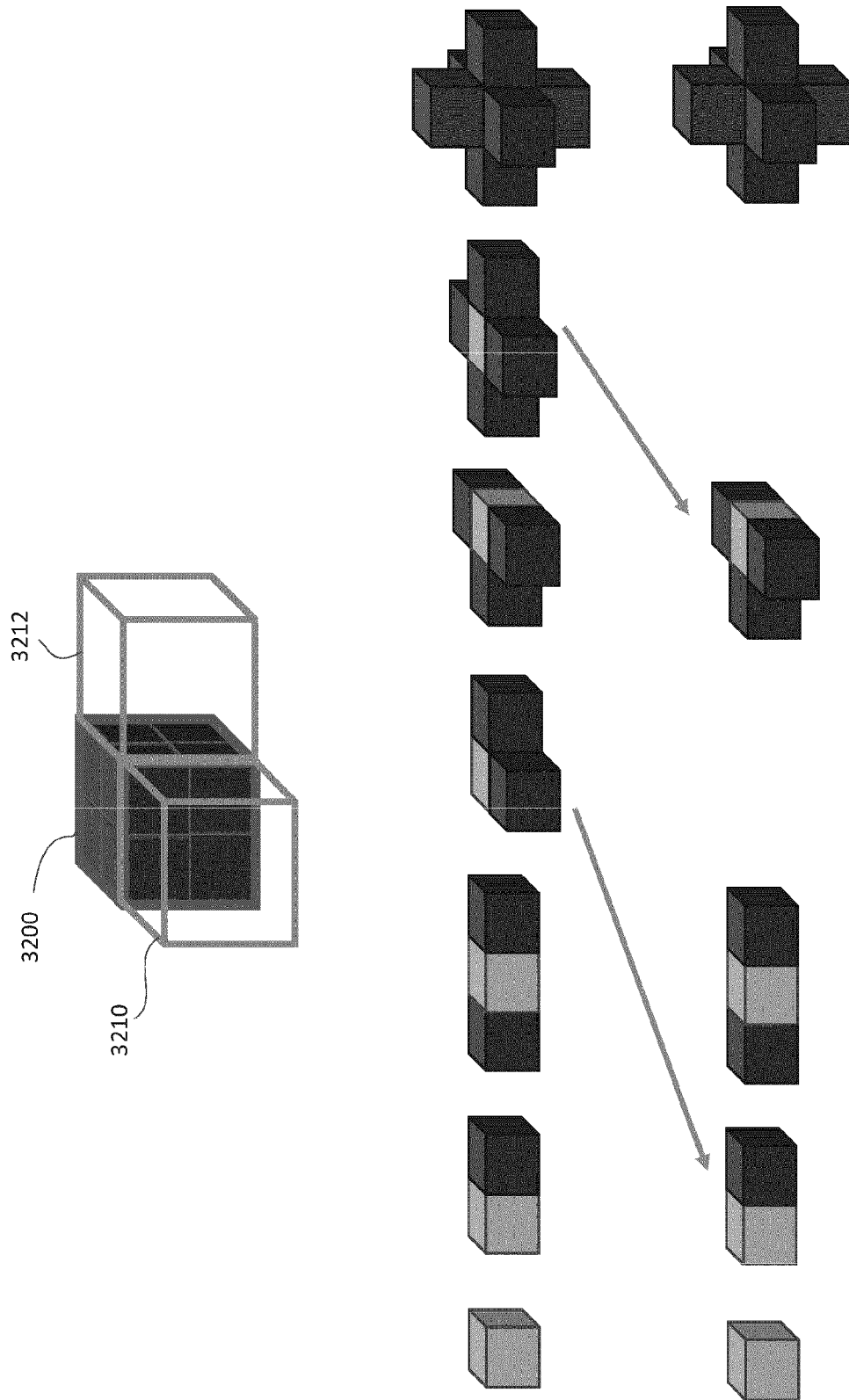
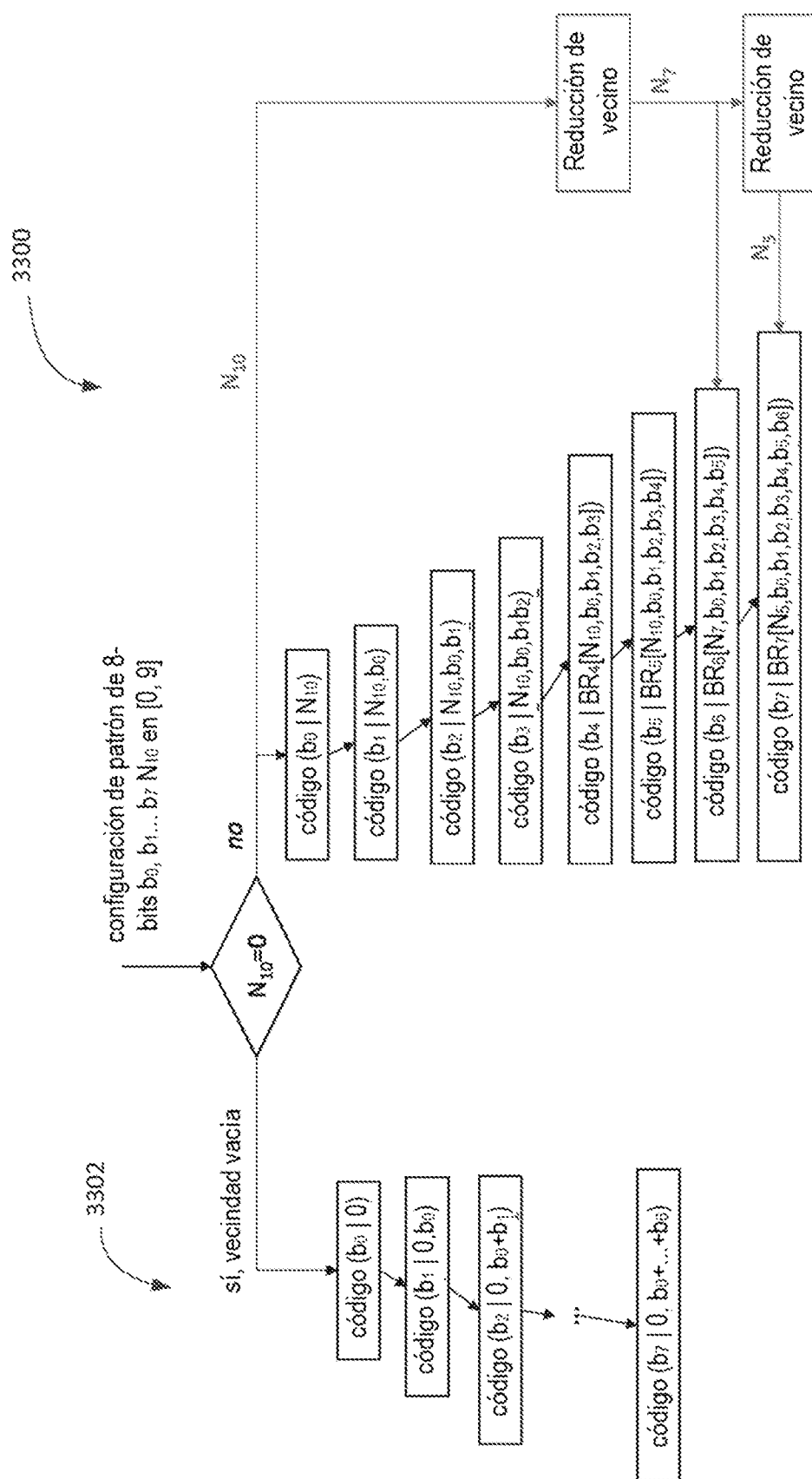


FIG. 25

**FIG. 26**

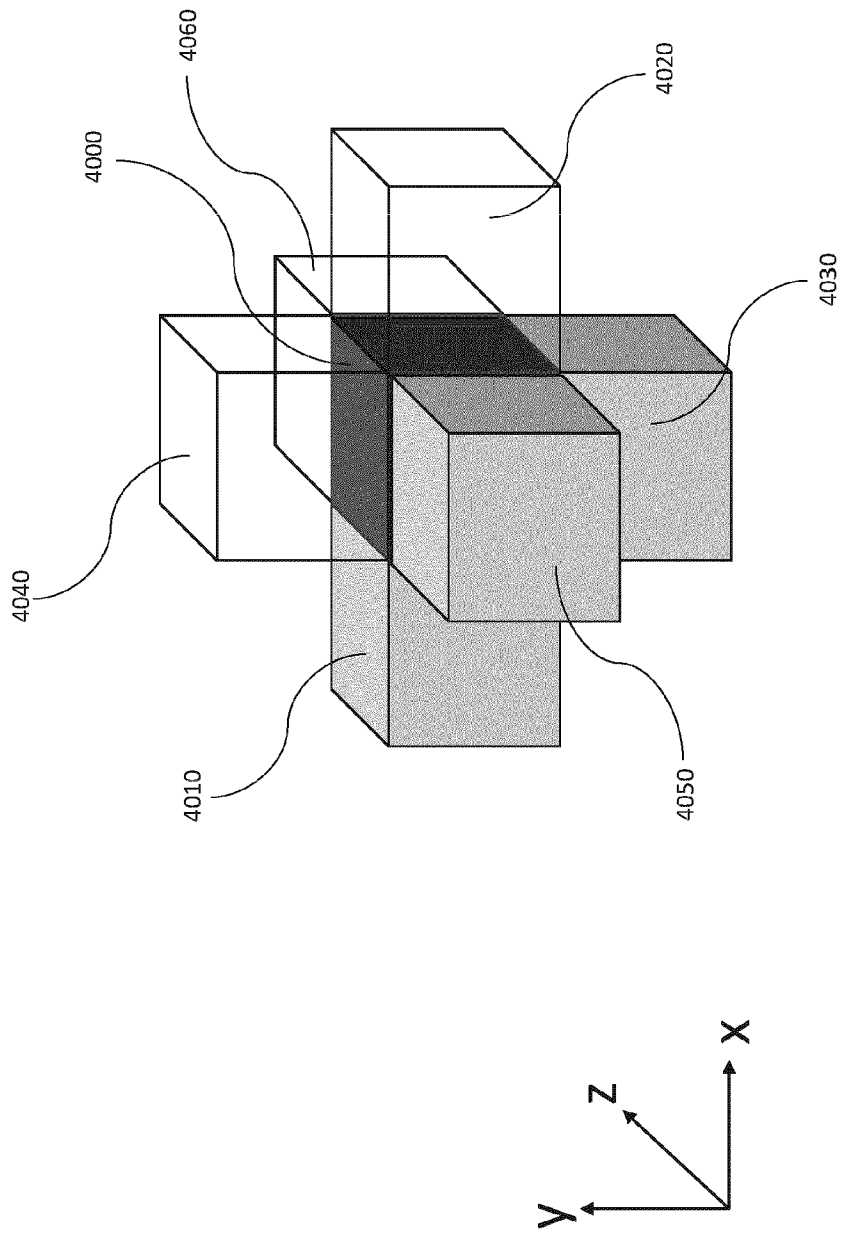


FIG. 27

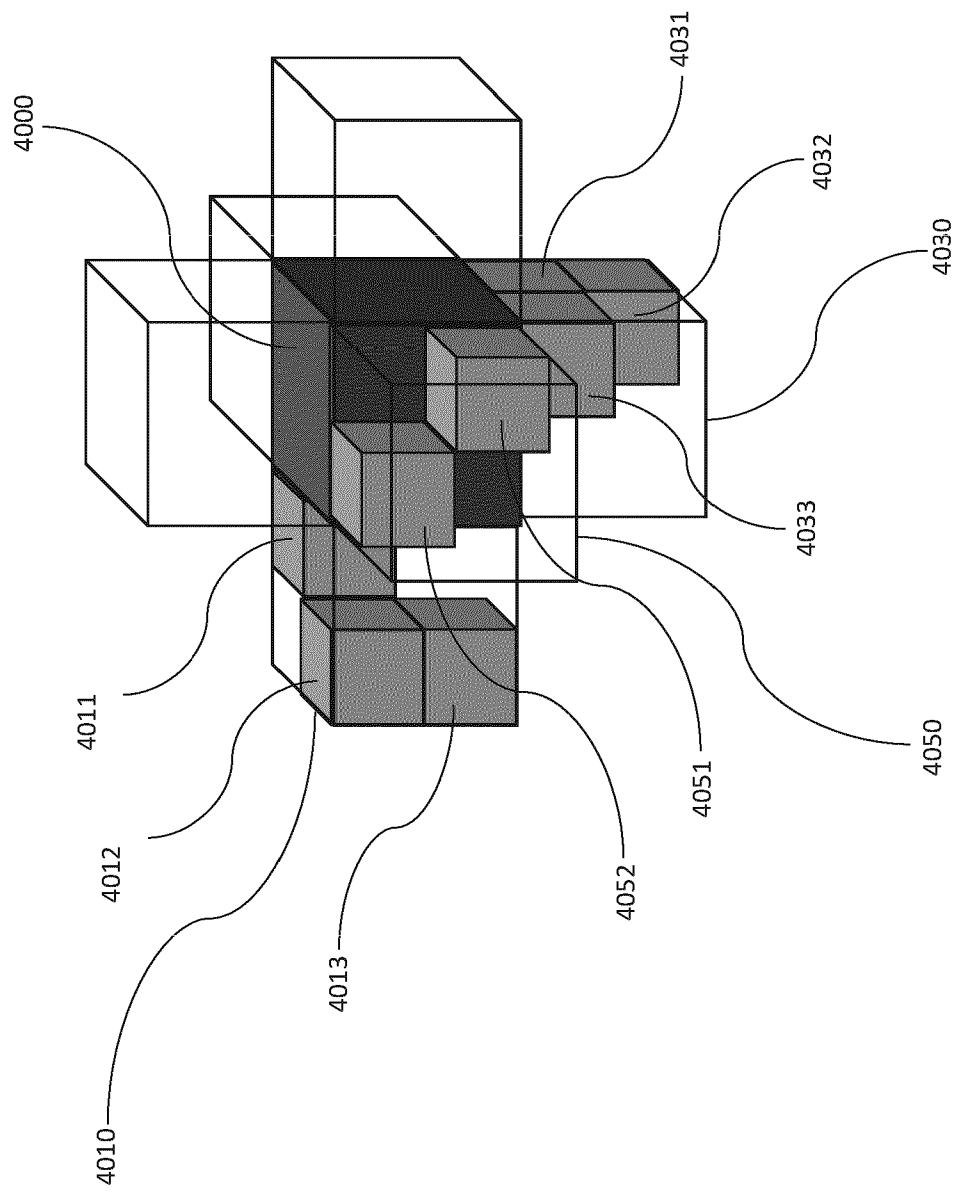
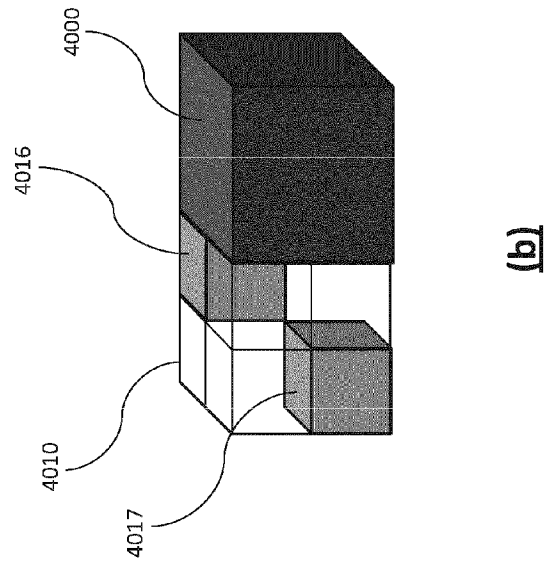
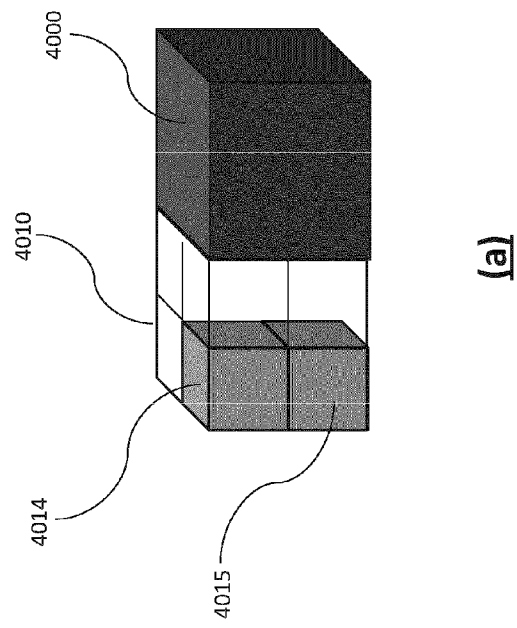


FIG. 28

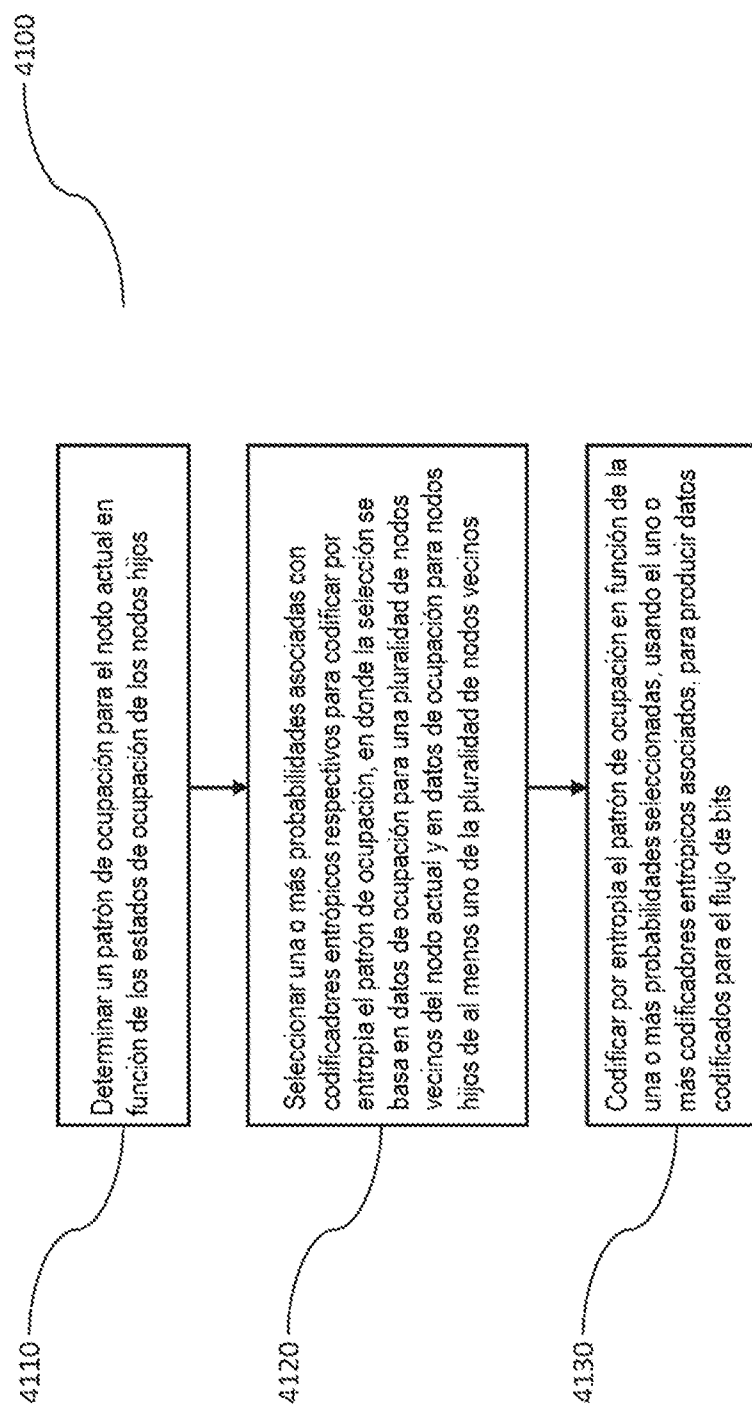


(b)

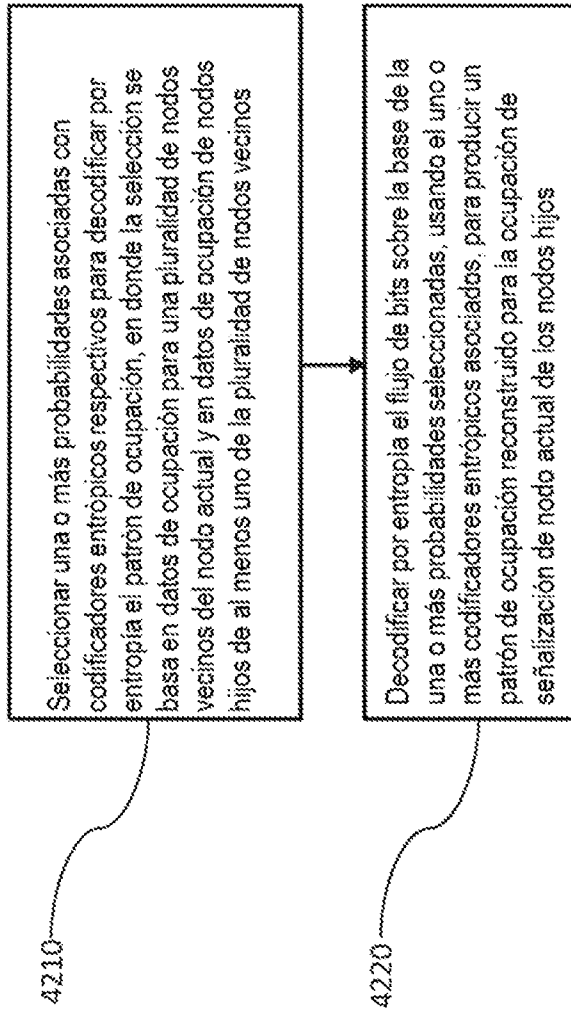


(a)

FIG. 29

**FIG. 30**

4200

**FIG. 31**

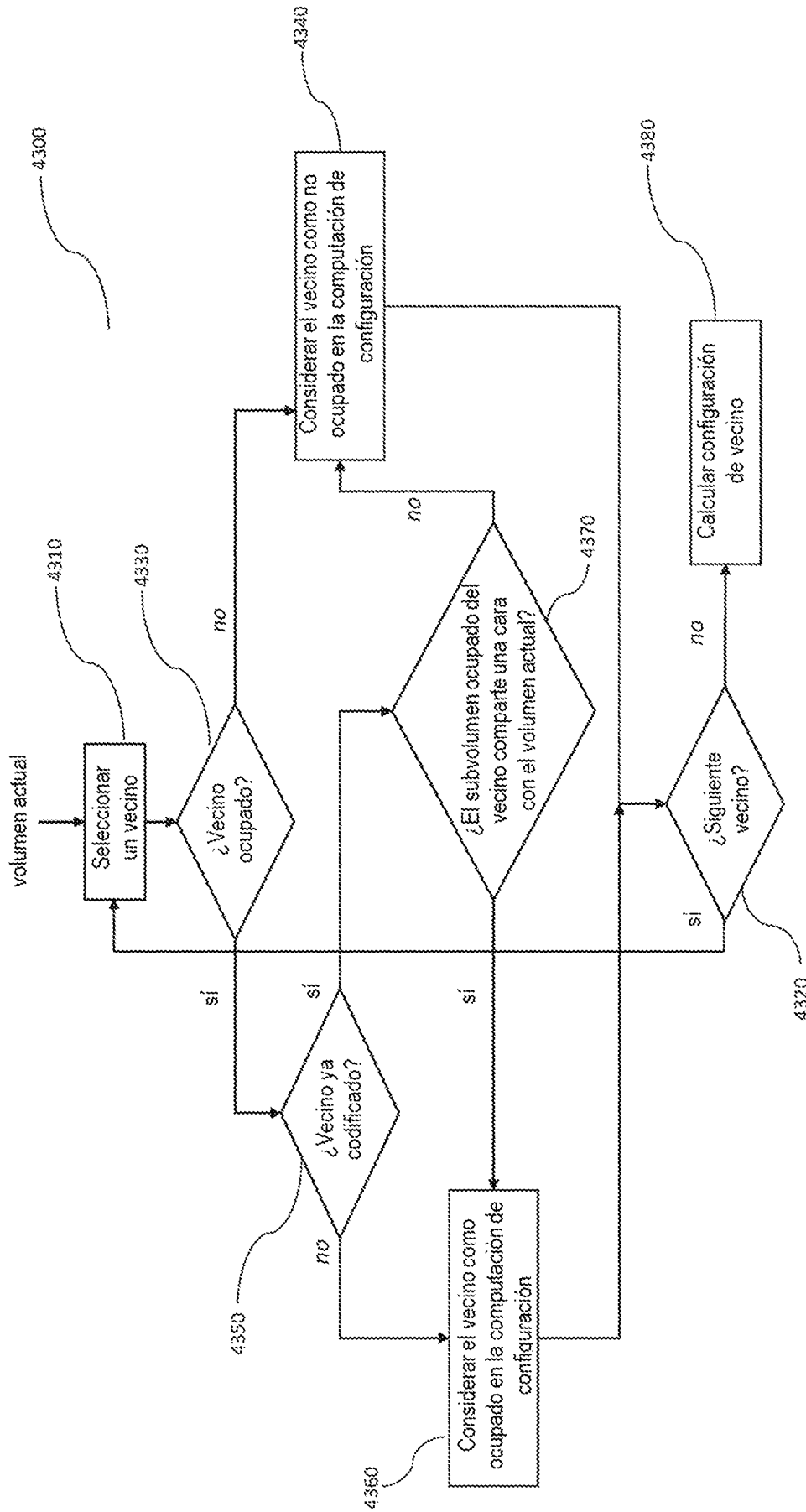
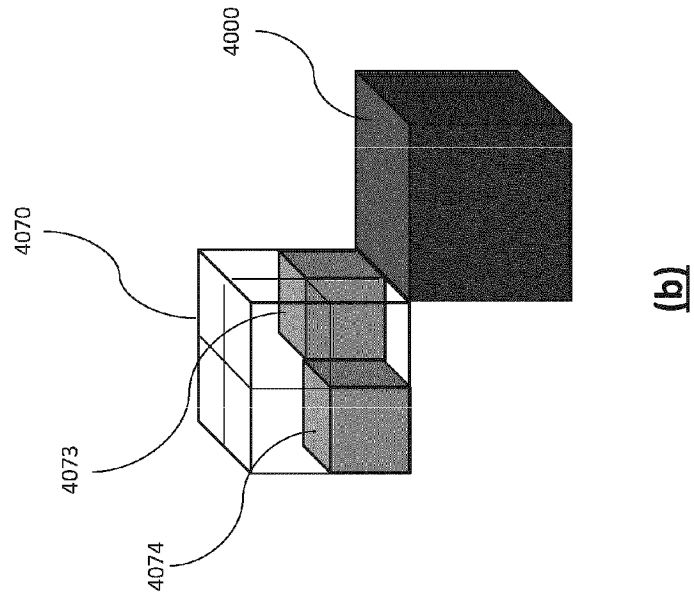
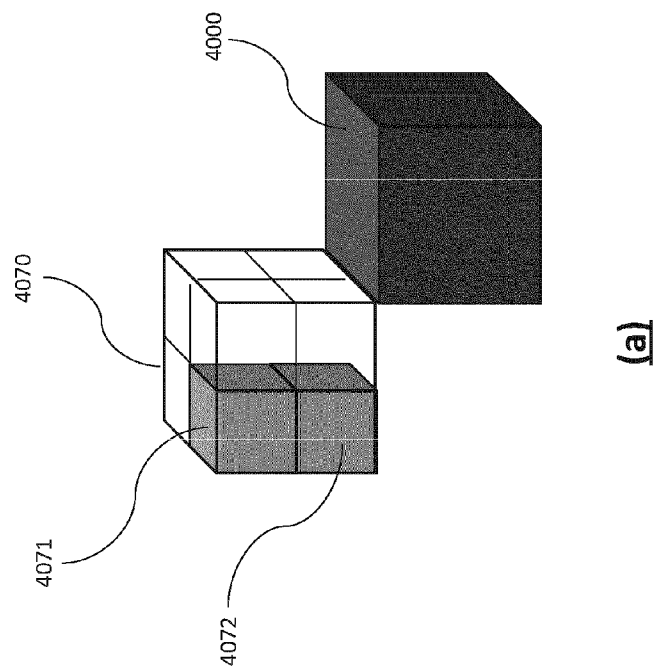


FIG. 32

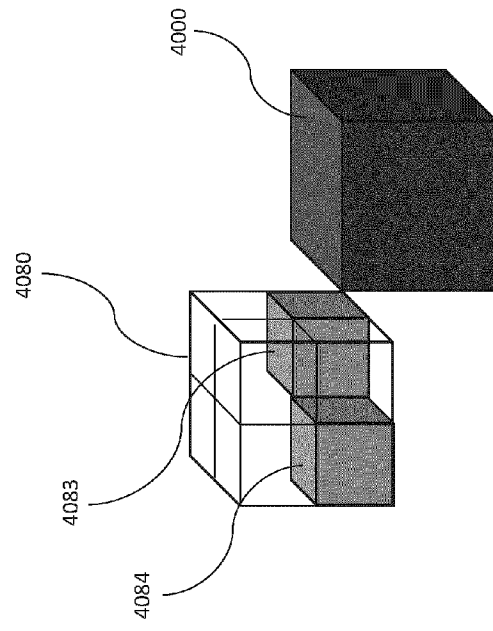


(b)

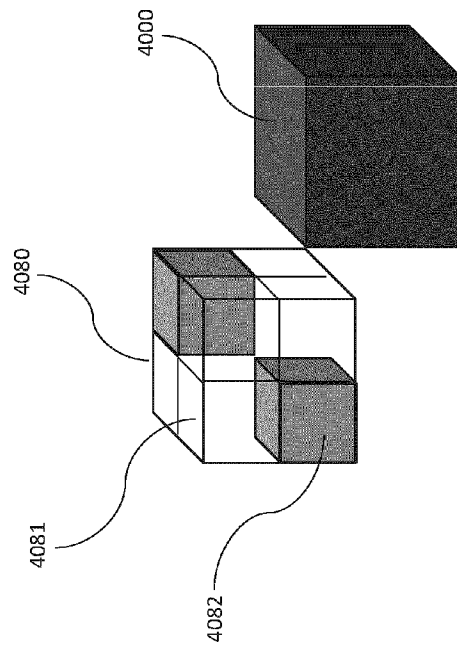


(a)

FIG. 33



(b)



(a)

FIG. 34