(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 381 777**

(51) Int. CI.:
H04N 7/24 (2011.01)
G06T 9/00 (2006.01)
H04N 7/26 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **02258162 .3**
(96) Fecha de presentación: **27.11.2002**
(97) Número de publicación de la solicitud: **1320264**
(97) Fecha de publicación de la solicitud: **18.06.2003**

(54) Título: **Procedimiento y aparato para codificar y decodificar un interpolador de orientación**

(30) Prioridad:
27.11.2001 US 333130 P
03.12.2001 US 334541 P
26.12.2001 US 342101 P
04.04.2002 US 369597 P
22.11.2002 KR 2002073044

(45) Fecha de publicación de la mención BOPI:
31.05.2012

(45) Fecha de la publicación del folleto de la patente:
31.05.2012

(73) Titular/es:
SAMSUNG ELECTRONICS CO., LTD.
416, MAETAN-DONG, PALDAL-GU
SUWON-CITY, KYUNGKI-DO, KR

(72) Inventor/es:
Kim, Do-Kyoon;
Jung, Seok-yoon;
Jang, Euee-seon;
Woo, Sang-oak;
Lee, Shin-jun;
Han, Mahn-jin y
Jang, Gyeong-ja

(74) Agente/Representante:
Carpintero López, Mario

DESCRIPCIÓN

Procedimiento y aparato para codificar y decodificar un interpolador de orientación

La presente invención se refiere a un procedimiento y un aparato para la codificación y decodificación de datos de animación tridimensionales, y más particularmente, a un procedimiento y un aparato para la codificación y decodificación de un interpolador de orientación que representa información sobre la rotación de un objeto en animación.

MPEG-4 BIFS, que es una de las normativas multimedia internacionales, soporta una animación basada en fotogramas clave usando un nodo interpolador que tiene las claves y los valores de claves de una animación.

Para representar la animación de forma natural y con suavidad siempre que sea posible usando tal técnica de animación basada en fotogramas clave, se requiere un número considerable de claves y una cantidad considerable de datos de valores de claves, y los datos de campo entre los fotogramas clave se rellenan por interpolación. La interpolación en un lenguaje de modelado de realidad virtual (VRML) involucra una interpolación lineal o esférica.

Las claves y los valores de las claves aproximan una curva de animación original sobre un eje temporal. La FIG. 1 es un diagrama que ilustra las trayectorias de dos dimensiones de los datos de animación, representados por un nodo interpolador de orientación, de acuerdo con el paso del tiempo sobre la superficie de una esfera tridimensional. Como se muestra en la FIG. 1, la normativa MPEG-4 BIFS convencional soporta la interpolación lineal esférica entre fotogramas clave y una trayectoria de animación parece similar a un conjunto de segmentos que representan la variación de los datos de animación.

En un nodo interpolador de orientación proporcionado por BIFS, los datos de claves indican un momento predeterminado de tiempo sobre un eje temporal donde está localizada una animación usando números discontinuos entre - ∞ y + ∞. Los datos de valores de claves representan la información sobre la rotación de un objeto en una imagen sintética en un momento predeterminado en el tiempo indicado por los datos de clave. La información sobre la rotación de un objeto en otro momento predeterminado de tiempo que no se representa por los datos claves, se obtiene usando los datos de claves correspondientes a dos momentos de tiempo, que son los más adyacentes al momento predeterminado en el tiempo, por interpolación lineal esférica.

En la interpolación lineal esférica, la información de rotación se representa por un eje de rotación y un ángulo de rotación. MPEG-4 BIFS, como el lenguaje de modelado de realidad virtual (VRLM), soporta la información de rotación representada por un eje de rotación y un ángulo de rotación usando un nodo interpolador de orientación. Cuando se genera una animación suave usando datos de valores de claves en la interpolación lineal esférica, los valores diferenciales de los datos de valores de claves entre los fotogramas clave están altamente correlacionados entre si, lo que causa redundancia entre los datos. Por consiguiente, es efectivo usar un procedimiento para la codificación de datos de valores de claves usando valores diferenciales de datos.

MPEG-4 BIFS proporciona dos procedimientos diferentes para la codificación de datos de campo representados por claves y datos de valores de claves de un nodo interpolador de orientación. Uno es el procedimiento para la codificación de los datos de campo usando la modulación de códigos de pulsos (PCM) y el otro es un procedimiento para la codificación de los datos de campo usando una modulación de códigos de pulsos diferenciales (DPCM) y la codificación de entropía.

En el procedimiento para la codificación de los datos de campo usando PCM, sólo se realiza un procedimiento de cuantización sobre los datos de clave y los datos de valores de claves a codificar. Como las características de los datos a codificar no se consideran en este procedimiento, este procedimiento se considera ineficaz. En el procedimiento para la codificación de los datos de campo usando PCM, se introducen los datos de campo de un nodo interpolador de orientación, y los datos de valores de claves de los datos de campo se convierten en valores de un espacio de cuaternios. A continuación, las claves y los datos de valores de las claves se cuantifican. Los datos de campo cuantizados se sacan en la forma de datos binarios. Para medir el grado en el que se visualizan distorsionados los resultados de la transformación de cuaternios en comparación con los datos de campo originales, los datos binarios se restauran en datos de valores de claves consistentes en un eje de rotación y un ángulo de rotación. Los datos de campo restaurados de un nodo interpolador de orientación se almacenan y a continuación se sacan sobre una pantalla, Es posible medir el grado de la distorsión visual de las imágenes causada por un error de cuaternios usando los datos restaurados. La distorsión de las imágenes puede calcularse con la ecuación (1) dada a continuación.

$$D = \sqrt{(\frac{\sum_{i=0}^{i<N} \varepsilon_i}{N})^2} = \sqrt{(\frac{\sum_{i=0}^{i<N} Q_i - \hat{Q}_i}{N})^2} \quad \ldots(1)$$

En la ecuación (1), N representa el número de datos de campo, y ε_i representa un valor diferencial entre los datos de valores de claves codificados Q_i y los datos del valor de clave $\hat{O}_i$ restaurado en un espacio de cuaternios.

Por otra parte, en el procedimiento para la codificación de los datos de campo usando DPCM y codificación de entropía, se considera una correlación entre datos sucesivos, y de este modo este procedimiento se considera más eficaz que el procedimiento para la codificación de los datos de campo usando PCM en términos de eficacia de la codificación. En este procedimiento, un valor diferencial entre los datos de valores de claves restauradas y los datos de valores de claves a codificar se calcula antes de un procedimiento de cuantización, y a continuación se cuantiza el valor diferencial, mejorando de este modo la eficacia de la codificación aprovechando las características de los datos mostrados en el valor diferencial.

Las FIG. 2A y 2B son diagramas de bloque de un codificador MPEG-4 PMFC que usa DPCM lineal y codificación de entropía, y un decodificador MPEG-4 PMFC que usa DPCM lineal inversa y decodificación de entropía, respectivamente. Un operador de DPCM lineal mostrado en la FIG. 2A calcula los datos diferenciales Q_i entre los datos de valores de claves actuales y los datos de valores de claves restaurados anteriormente siguiendo la ecuación (2).

$$\dot{Q}_i = Q_i - \hat{Q}_{i-1} = (q_{i,0} - \hat{q}_{i-1,0}, q_{i,1} - \hat{q}_{i-1,1}, q_{i,2} - \hat{q}_{i-2,2}, q_{i,3} - \hat{q}_{i-1,3}) \quad \ldots(2)$$

En la ecuación (2), Q_i representa los datos de valores de claves originales en un momento predeterminado en el tiempo (t), que se representan por un cuaternio, y $\hat{Q}_{i-1}$ representa los datos de valores de claves en un momento predeterminado en el tiempo (t – 1), que se restauran a partir de un circuito de compensación de errores.

Sin embargo, el procedimiento de la codificación realizado en el aparato para la codificación de los datos de valores de claves mostrado en la FIG. 2A no tiene una alta eficacia de codificación. Es posible deducir fácilmente cuáles son los inconvenientes del procedimiento de la codificación analizando los datos de valores de claves, que determinan la rotación de un objeto en un espacio de cuaternios. Los datos de valores de claves se representan por un cuaternio en la siguiente ecuación:

$$Q = (\cos\frac{\theta}{2}, \frac{n_x}{\|n\|}\text{sen}\,\frac{\theta}{2}, \frac{n_y}{\|n\|}\text{sen}\,\frac{\theta}{2}, \frac{n_z}{\|n\|}\text{sen}\,\frac{\theta}{2}) \quad \cdots(3) \quad \ldots(3)$$

Por ejemplo, cuando las componentes de un cuaternio tienen los mismos valores absolutos que sus componentes correspondientes de otro cuaternio pero diferentes signos en un espacio de cuaternios, como se muestra en la Ecuación (3), los dos cuaternios se consideran los mismos. En otras palabras, los dos cuaternios proporcionan los mismos efectos en términos de transformación rotacional de un objeto en un espacio de 3D, lo que significa que los factores que afectan a la transformación rotacional de un objeto son una dirección de un eje de rotación y un ángulo de rotación, más que el vector del eje de rotación. Sin embargo, como en MPEG-4 BIFS, si los datos de valores de claves se representan por un cuaternio usando la ecuación (3) y se calcula un valor diferencial de forma lineal calculando las diferencias en los vectores entre datos de valores de claves sucesivos, el valor diferencia es distinto de 0, lo que significa que los valores diferenciales lineales no reflejan bien la redundancia en la transformación rotacional. Por consiguiente, es imposible medir con precisión la calidad de las imágenes usando el procedimiento para medir el grado de distorsión de las imágenes mostrado en la ecuación (1).

Un procedimiento de una compresión de movimiento compensado de modelos de animación en 3D se trata en el documento "Compresión de movimiento compensado de modelos de animación en 3D" de Ahn, Kim, Kuo y Ho, IEEE Electronics Letters, Volumen 37, Nº 24, páginas 1445-1446.

MPEG ha emitido una llamada para propuestas para la compresión de interpoladores y publicó un conjunto de condiciones experimentales para la evaluación del funcionamiento en esta tarea (ISO/IEC JTC1/SC29/WG11 Documentos de Salida N4098 y N4364, respectivamente).

La Normativa Internacional ISO/IEC 14496-1, Primera Edición (Tecnología de la Información – Codificación de objetos audio – visuales – Parte 1: Sistemas) especifica las funcionalidades a nivel del sistema para la comunicación de escenas interactivas audio – visuales.

El documento WO 01/41156 describe un procedimiento de procesamiento de un fichero de datos que especifica una jerarquía de nodos, para determinar un intervalo de parámetros asumido por los nodos.

El documento US 6.075.901 desvela un procedimiento y un sistema para codificar de forma uniforme disposiciones de valores en un flujo de video.

Una historia de la codificación de animación en 3D se presenta en el documento "Codificación de la animación de 3D: su historia y estructura", de Jang, Procedimientos de IEEE Conferencia Internacional sobre Multimedia y Exposición 2000, Volumen 2, páginas 1119 – 1122.

Un análisis de la normativa de MPEG-4 se presenta en el documento "Formato Binario para la Escena (BIFS): Combinación de medios MPEG-4 para construir servicios ricos en multimedia" de Signes, Procedimientos de SPIE, Volumen 3653, páginas 1506 – 1517.

La presente invención busca proporcionar un procedimiento y un aparato para codificar y decodificar un interpolador de orientación, que codifica y decodifica un interpolador de orientación extraído constituido por puntos de ruptura extraídos de un interpolador de orientación original de modo que impide que un error entre el interpolador de orientación extraído y el interpolador de orientación original sea mayor que un límite de error permisible y de este modo puede proporcionar una animación de alta calidad con una tasa de compresión elevada.

Es otro aspecto de la presente invención proporcionar un procedimiento y un aparato para la codificación y decodificación de un interpolador de orientación, lo que puede proporcionar una animación de alta calidad con una tasa de compresión elevada calculando un valor diferencial rotacional, que puede reflejar de forma suficiente la redundancia en la transformación rotacional, y codificar los datos de valores de claves de un interpolador de orientación usando el valor diferencial rotacional.

Es otro aspecto de la presente invención proporcionar un flujo de bits codificado y decodificado por un procedimiento y un aparato para la codificación y decodificación de un interpolador de orientación de acuerdo con la presente invención, que puede proporcionar una animación de alta calidad con una tasa de compresión elevada.

Por consiguiente, para conseguir los anteriores así como otros aspectos de la presente invención se proporciona un aparato para la codificación de un interpolador de orientación de acuerdo con la reivindicación 1.

Preferiblemente, el aparato incluye además un dispositivo de re-muestreo que muestrea la primera trayectoria de animación en un número predeterminado de secciones que tienen un intervalo de una magnitud predeterminada de tiempo y saca un interpolador de orientación incluyendo los datos de claves re-muestreados y los datos de valores de claves re-muestreados, y un selector que saca una entrada del interpolador de orientación dentro del mismo al dispositivo de re-muestreo o al extractor de puntos de ruptura en respuesta a una señal de entrada externa.

Para conseguir lo anterior así como otros aspectos de la presente invención, se proporciona un aparato para la codificación de un interpolador de orientación que incluye datos de claves que indican las localizaciones de los fotogramas clave sobre un eje temporal y los datos de valores de claves que indican la rotación de un objeto. El aparato incluye un dispositivo de re-muestreo que muestra una trayectoria de animación constituida por un interpolador de orientación de entrada dentro de un número predeterminado de secciones que tienen un intervalo de una cantidad predeterminada de tiempo y saca un interpolador de orientación incluyendo los datos de claves re-muestreados y los datos de valores de claves re-muestreados, un codificador de datos de claves que codifica la entrada de datos de claves del dispositivo de re-muestreo, y un codificador de datos de valores de claves que genera un valor diferencial rotacional usado para girar un objeto tanto como la diferencia entre la transformación rotacional aplicada al objeto por los datos de valores de claves de un fotograma clave actual y la transformación rotacional aplicada al objeto por los datos de valores de claves de un fotograma clave anterior y de este modo codifica la entrada de datos de valores de claves desde el dispositivo de re-muestreo.

Preferiblemente, el extractor de puntos de ruptura incluye un interpolador lineal que extrae un punto de comienzo de la trayectoria y un punto de final de la trayectoria de una trayectoria de animación de la entrada, selecciona puntos de la trayectoria entre los puntos de la trayectoria de comienzo y de final e interpola otros puntos de la trayectoria, que aún no se han seleccionado, usando los puntos de la trayectoria seleccionados, un calculador de error que calcula un error entre la trayectoria de animación de entrada y una trayectoria de animación interpolada generada por el interpolador lineal usando la interpolación, y una unidad de determinación que extrae puntos de ruptura, por lo que un error entre la trayectoria de animación de entrada y la trayectoria de animación interpolada puede minimizarse, y saca los puntos de ruptura seleccionados si el error correspondiente no es mayor que un límite de error predeterminado.

Preferiblemente, el codificador de valores de claves incluye un generador de datos diferenciales rotacionales que genera, usando un valor de la transformación rotacional de un fotograma clave actual y un valor de la transformación rotacional restaurado de un fotograma clave anterior, un valor diferencial rotacional usado para girar el objeto tanto como la diferencia entre la transformación rotacional aplicada al objeto en el fotograma clave actual por los datos de valores de claves, y la transformación rotacional aplicada al objeto en el fotograma clave anterior por los datos de valores de claves, y saca los datos diferenciales rotacionales por la cuantización del valor diferencial rotacional, y un codificador de entropía que codifica la entropía de los datos diferenciales rotacionales.

Preferiblemente, el generador de datos diferenciales rotacionales incluye un primer multiplicador de cuaternios que genera el valor diferencial rotacional usando el valor de la transformación rotacional del fotograma clave actual y el valor de la transformación rotacional restaurado del fotograma clave anterior, un dispositivo de cuantización que genera los datos diferenciales rotacionales por la cuantización de los valores diferenciales rotacionales, y un

dispositivo de cuantización inversa que genera un valor diferencial rotacional restaurado por la cuantización inversa de los datos diferenciales rotacionales, y un segundo multiplicador de cuaternios que genera un valor de la transformación rotacional restaurado del fotograma clave actual por la multiplicación de los cuaternios del valor diferencial rotacional restaurado por un valor de la transformación rotacional del fotograma clave anterior.

Preferiblemente, el codificador de datos de claves incluye un primer dispositivo de cuantización que cuantiza los datos de claves de un interpolador de orientación que usa bits de cuantización predeterminados, un primer procesador de DPCM que genera datos diferenciales de los datos de claves cuantizados, un procesador DND que realiza una operación de DND sobre los datos diferenciales dependiendo de una relación entre los datos diferenciales y un valor máximo y un valor mínimo entre ellos, y un primer codificador de entropía que codifica la entropía de la entrada de datos diferencial desde el procesador de DND.

Para conseguir lo anterior así como otros aspectos de la presente invención se proporciona un aparato para decodificar un flujo de bits de acuerdo con la reivindicación 17.

Preferiblemente el decodificador de datos de valores de claves incluye un decodificador de entropía que genera datos diferenciales rotacionales de la DPCM circular o datos diferenciales rotacionales cuantizados por la decodificación de entropía de los datos de valores de claves a partir del flujo de bits, un operador de la DPCM circular inversa que genera datos diferenciales rotacionales cuantizados realizando la operación de la DPCM circular inversa sobre la entrada de datos diferenciales rotacionales desde el decodificador de entropía que sigue el orden de la operación de DPCM decodificados desde el flujo de bits, un dispositivo de cuantización inversa que gira el objeto tanto como la diferencia entre las transformaciones rotacionales aplicadas al objeto por los datos de valores de claves de cuaternios de cada uno de los fotogramas clave por la cuantización inversa de los datos diferenciales rotacionales cuantizados, y un multiplicador de cuaternios que genera un valor de la transformación rotacional de un fotograma clave actual por multiplicación de cuaternios de un valor diferencial rotacional del fotograma clave actual por el valor de la transformación rotacional restaurado de un fotograma clave anterior.

Para conseguir lo anterior así como otros aspectos de la presente invención, se proporciona un procedimiento para la codificación de un interpolador de orientación de acuerdo con la reivindicación 21.

Preferiblemente, la etapa (b) incluye (b1) seleccionar un punto de comienzo de la trayectoria y un punto de final de la trayectoria de la primera trayectoria de animación, (b2) seleccionar puntos de la trayectoria entre los puntos de comienzo y de final de la trayectoria uno por uno e interpolar los otros puntos de la trayectoria, que aún no se han seleccionado, usando los puntos de la trayectoria seleccionados, (b3) calcular un error entre la primera trayectoria de animación y una segunda trayectoria de animación generada por la interpolación en la etapa (b2), y (b4) seleccionando los puntos de ruptura por los cuales puede minimizarse el error entre la primera trayectoria de animación y la segunda trayectoria de animación, comprobando si el error correspondiente no es mayor de un límite de error predeterminado y determinar los datos de claves y los datos de valores de claves a codificar.

Preferiblemente, el procedimiento para la codificación de un interpolador de orientación puede incluir además (a) generar un interpolador de orientación incluyendo datos de claves re-muestreados y datos de valores de claves re-muestreados muestreando la primera trayectoria de animación dentro de un número predeterminado de secciones que tienen un intervalo de magnitud predeterminada de tiempo, antes de la etapa (b) o puede incluir además (c) generar datos de claves y datos de valores de claves a codificar muestreando la segunda trayectoria de animación constituida usando los puntos de ruptura extraídos en un número predeterminado de secciones que tienen un intervalo de un número predeterminado de tiempo, después de la etapa (b).

Preferiblemente, la etapa (d) incluye la cuantización de los datos de claves con un número predeterminado de bits de cuantización, generando los datos diferenciales realizando una operación de DPCM predeterminada sobre los datos cuantizados, y decodificando la entropía de los datos diferenciales.

Preferiblemente, la etapa (e) incluye (e1) generar un valor diferencial rotacional usado para girar el objeto tanto como la diferencia entre las transformaciones rotacionales aplicadas al objeto por los datos de valores de claves de los fotogramas clave actual y anterior usando un valor de la transformación rotacional del fotograma clave actual y un valor de la transformación rotacional restaurado del fotograma clave anterior y generar los datos diferenciales rotacionales por la cuantización de los valores diferenciales rotacionales, (e2) realizar de forma selectiva una operación de DPCM lineal o una operación de DPCM circular sobre los datos diferenciales rotacionales, y (e3) codificar la entropía de los datos diferenciales rotacionales.

Preferiblemente, la etapa (e1) incluye (e11) generar el valor diferencial rotacional usando un valor de la transformación rotacional del fotograma clave actual y un valor de transformación rotacional restaurado del fotograma clave anterior, (e12) generar los datos diferenciales rotacionales cuantificando el valor diferencial rotacional, (e13) generar un valor diferencial rotacional restaurado por la cuantización inversa de los datos diferenciales rotacionales y (e14) generar un valor de la transformación rotacional restaurado del fotograma clave actual por la multiplicación de cuaternios del valor diferencial rotacional restaurado por un valor de la transformación rotacional restaurado del fotograma clave anterior.

Preferiblemente, la etapa (a) incluye generar los datos diferenciales por decodificación de entropía del flujo de bits de entrada, generar los datos de claves cuantizados realizando una operación de DPCM predeterminada y una operación de DND inversa sobre los datos diferenciales, y generar datos de claves restaurados por la cuantización inversa de los datos de valores de claves cuantizados.

Preferiblemente, la etapa (b) incluye (b1) la generación de datos diferenciales rotacionales con la DPCM circular o datos diferenciales rotacionales cuantizados por la decodificación de entropía de los datos de valores de claves a partir del flujo de bits, (b2) la generación de datos diferenciales rotacionales realizando una operación de DPCM circular inversa sobre los datos diferenciales rotacionales decodificados en entropía que siguen el orden de la operación de DPCM decodificados a partir del flujo de bits, (b3) la generación de un valor diferencial rotacional usado para girar el objeto tanto como la diferencia entre las transformaciones rotacionales aplicadas al objeto por los datos de valores de claves de cuaternios de cada uno de los fotogramas clave por la cuantización inversa de los datos diferenciales rotacionales, y (b4) generar un valor de la transformación rotacional de un fotograma clave actual multiplicando los cuaternios de un valor diferencial rotacional del fotograma clave actual por un valor de la transformación rotacional decodificado de un fotograma clave anterior.

Para conseguir lo anterior así como otros aspectos de la presente invención, se proporciona un flujo de bits de acuerdo con la reivindicación 33. En ese punto, la información de codificación/decodificación de los datos de claves incluye información de la operación de DND inversa que incluye el orden de DND inversa que indica un número predeterminado de ciclos de la DND inversa a realizar sobre los datos diferenciales generados por la decodificación de entropía del flujo de bits para extender el intervalo de los datos diferenciales y los valores máximo y mínimo de entre los datos diferenciales usados en cada uno de los ciclos de la operación de DND inversa, en primer lugar la información de la operación de DPCM inversa incluyendo el orden de la operación de DPCM inversa a realizar sobre los datos diferenciales con la DND inversa de modo que transforma los datos diferenciales con la DND inversa en datos de claves cuantizados y datos de intra claves que se usan para cada uno de los ciclos de la operación de DPCM inversa, y la primera información de cuantización inversa usada en la cuantización inversa para generar datos de claves restaurados por la cuantización inversa de los datos de claves cuantizados. Los datos de valores de claves de la información de codificación/decodificación incluyen los datos diferenciales rotacionales codificados en entropía por la cuantización de un valor diferencial rotacional usados para girar el objeto tanto como la diferencia entre las transformaciones rotacionales aplicadas al objeto por los datos de valores de claves de cuaternios de cada uno de los fotograma clave, la información de decodificación de entropía incluyendo un modo de decodificación de entropía que indica un procedimiento de decodificación de entropía a realizar sobre los datos diferenciales rotacionales, la información de la operación de DPCM circular inversa incluyendo el orden de la operación de DPCM circular inversa, que indica si la operación de DPCM circular inversa se realizará o no sobre los datos diferenciales rotacionales decodificados por entropía siguiendo el modo de decodificación de entropía, y una segunda información de cuantización inversa incluyendo un número predeterminado de bits de cuantización inversa usados para restaurar los datos de valores de clave originales por la cuantización inversa de los datos de valores de claves cuantizados.

Preferiblemente, la información de codificación/decodificación de datos de claves incluye además la información de decodificación de claves lineal usada para la decodificación de una región de claves lineal incluida en el flujo de bits, y la información de decodificación de claves lineal incluye un indicador que indica si existe o no la región de claves lineal donde los datos de claves aumentan de forma lineal de entre los datos de claves, incluido el número de datos de claves en la región de claves lineal, y los datos de claves de comienzo y de final de la región de claves lineal.

Los objetos anteriores y las ventajas de la presente invención se harán más evidentes por la descripción en detalle de las realizaciones preferidas con referencia a los dibujos adjuntos en los que:

la FIG. 1 es un diagrama que ilustra trayectorias de dos dimensiones de datos de animación, representados por un nodo interpolador de orientación, de acuerdo con el paso del tiempo sobre la superficie de una esfera de tres dimensiones;

las FIG. 2A y 2B son diagramas de bloques de un codificador MPEG-4 PMFC que usa DPCM lineal y codificación de entropía y un decodificador MPEG-4 PMFC que usa DPCM lineal inversa y decodificación de entropía, respectivamente;

la FIG. 3A es un diagrama de bloques de un aparato para la codificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención, y la FIG. 3B es un diagrama de flujo de un procedimiento para la codificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención;

las FIG. 4A hasta 4C son diagramas de bloque de ejemplos de un analizador de acuerdo con las realizaciones preferidas de la presente invención;

la FIG. 5A es un diagrama de flujo de una etapa S320 en la FIG. 3B;

la FIG. 5B es un diagrama de flujo de un procedimiento de re-muestreo de acuerdo con una realización preferida de la presente invención;

la FIG. 5C es un diagrama de flujo de un procedimiento para la extracción de puntos de ruptura de acuerdo con una realización preferida de la presente invención;

la FIG. 6A es un diagrama que ilustra datos de claves originales y datos de claves re-muestreados, y la FIG. 6B es un diagrama que ilustra una trayectoria de animación original y una trayectoria de animación re-muestreada;

las FIG. 7A hasta 7F son diagramas que ilustran un ejemplo de un procedimiento para la extracción de puntos de ruptura de acuerdo con una realización preferida de la presente invención;
la FIG. 8 es un diagrama que ilustra la salida de los datos de claves y los datos de valores de claves desde un extractor de puntos de ruptura en una configuración de generación de puntos de ruptura;
la FIG. 9A es un diagrama de bloques de un codificador de datos de claves de acuerdo con una realización preferida de la presente invención;
la FIG. 9B es un diagrama de bloques de un procesador de DND mostrado en la FIG. 9A;
las FIG. 10A hasta 10E son diagramas de flujo de un procedimiento de codificación de datos de claves de acuerdo con una realización preferida de la presente invención;
la FIG. 11 es un diagrama que ilustra un ejemplo de una función encodeSignedAAC;
las FIG.12A hasta 12J son diagramas que ilustran datos de claves obtenidos después de la realización de diferentes etapas de la codificación de datos de claves de acuerdo con una realización preferida de la presente invención;
la FIG. 13A es un diagrama de bloques de un codificador de datos de valores de claves de acuerdo con una primera realización de la presente invención, y la FIG. 13B es un diagrama de flujo de un procedimiento de codificación de datos de valores de claves de acuerdo con una primera realización de la presente invención;
la FIG. 14A es un diagrama que ilustra un ejemplo típico de una función de distribución de probabilidad (PDF) en cada una de las componentes de un valor diferencial rotacional;
la FIG. 14B es una curva arco tangente para una cuantización no lineal;
la FIG. 15A es un ejemplo de una salida de datos diferenciales rotacionales desde un dispositivo de cuantización incluido en un codificador de datos de valores de claves de acuerdo con una realización preferida de la presente invención; la FIG. 15B es un diagrama que ilustra los resultados de la realización de una operación de DPCM circular sobre los datos diferenciales mostrados en la FIG. 15A, y la FIG. 15C es un diagrama que ilustra los resultados de la realización de una operación de DPCM circular sobre datos diferenciales con la DPCM lineal mostrados en la FIG. 15B;
la FIG. 16 es un diagrama que ilustra un ejemplo de una función UnaryAAC () usada para la codificación de entropía;
la FIG. 17 es un diagrama que ilustra un error de la dirección de giro que ocurre durante la codificación de los valores de cuaternios de la transformación rotacional usando un valor diferencial rotacional;
la FIG. 18A es un diagrama de bloques de un operador de DPCM rotacional de un codificador de datos de valores de claves de acuerdo con una segunda realización de la presente invención, y la FIG. 18B es un diagrama de bloques de un calculador de errores de la dirección de giro mostrado en la FIG. 18A;
la FIG. 19A es un diagrama de flujo de una operación de DPCM rotacional de acuerdo con una segunda realización de la presente invención, y la FIG. 19B es un diagrama de flujo que ilustra las operaciones de un calculador de errores de la dirección de giro, un detector de errores de la dirección de giro y un corrector de la dirección de giro mostrados en la FIG. 9A;
la FIG. 20A es un diagrama de bloques de un dispositivo de cuantización de un codificador de datos de valores de claves para una tercera realización de la presente invención, y la FIG. 20B es un diagrama de flujo del funcionamiento de un dispositivo de cuantización de acuerdo con una tercera realización de la presente invención;
la FIG. 21A es un diagrama de bloques de un aparato para la decodificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención, y la FIG. 11B es un diagrama de flujo de un procedimiento para la decodificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención;
la FIG. 22 es un diagrama de bloques de un decodificador de datos de claves de acuerdo con una realización preferida de la presente invención;
las FIG. 23A y 23B son diagramas de flujo de un procedimiento de decodificación de datos de claves de acuerdo con una realización preferida de la presente invención;
la FIG. 24A es un diagrama de bloques de un decodificador de datos de valores de claves de acuerdo con una realización preferida de la presente invención, y la FIG. 24B es un diagrama de flujo de un procedimiento de codificación de datos de valores de claves de acuerdo con una realización preferida de la presente invención;
la FIG. 25 es un diagrama que ilustra la estructura de una entrada de un flujo de bits en un decodificador de entropía de un decodificador de datos de valores de claves de acuerdo con una realización preferida de la presente invención;
la FIG. 26 es un diagrama de flujo de un procedimiento para la síntesis de datos de claves y de datos de valores de claves de un interpolador de orientación de acuerdo con una realización preferida de la presente invención;
la FIG. 27 es un diagrama que ilustra un ejemplo de un procedimiento de cálculo de un error entre un interpolador de orientación a codificar y un interpolador de orientación decodificado;
la FIG. 28 es un diagrama para la comparación del funcionamiento de un procedimiento para la codificación de un interpolador de orientación de acuerdo con la presente invención con el funcionamiento de un procedimiento convencional para la codificación de un interpolador de orientación; y
las FIG. 29A hasta 29J son códigos de programa de lenguaje SDL, por los cuales se realiza un aparato para la decodificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención, que decodifica los datos de claves y los datos de valores de claves.

Se describirán un aparato y un procedimiento para la codificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención de forma más completa con referencia los dibujos adjuntos.

La FIG. 3A es un diagrama de bloques de un aparato para la codificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención.

Refiriéndonos a la FIG. 3A, el aparato para la codificación de un interpolador de orientación incluye un analizador 40, un codificador de datos de clave 200, un codificador de datos de valores de claves 300, y un codificador de cabecera 400.

La FIG. 3B es un diagrama de flujo de un procedimiento para la codificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención. Refiriéndonos a la FIG. 3B, se introduce un interpolador de orientación para su codificación dentro del analizador 40 en la etapa S300. En la etapa S320, el analizador 40 extrae los datos de claves y los datos de valores de claves a codificar desde una primera trayectoria de animación comprendida de datos de valores de claves de las componentes x, y, z y θ (ángulo de rotación) del interpolador de orientación, saca los datos de claves extraídos al codificador de datos de claves 200, y saca los datos de valores de claves extraídos al codificador de datos de valores de claves 300.

El codificador de datos de claves 200 realiza la cuantización de la entrada de datos de claves desde el analizador 40 usando un número predeterminado de bits de cuantización, genera datos diferenciales realizando una operación de DPCM predeterminada sobre los datos de claves cuantizados, y codifica la entropía de los datos diferenciales en la etapa S340.

El codificador de datos de valores de claves 300 realiza la cuantización de la entrada de datos de valores de claves desde el analizador 40, usando un número predeterminado de bits de cuantización, genera datos diferenciales realizando una operación de DPCM predeterminada sobre los datos cuantizados, y codifica los datos diferenciales en la etapa S360.

El codificador de cabecera 400 recibe la información necesaria para decodificar los datos de claves y los datos de valores de claves desde el codificador de datos de claves 200 y el codificador de datos de valores de claves 300 y codifica la información en la etapa S380.

En lo sucesivo, se describirán las estructuras y las operaciones del analizador 40, el codificador de datos de claves 200, y el codificador de datos de valores de claves 300 con mayor detalle con los dibujos adjuntos.

La FIG. 4A es un diagrama de bloques de un ejemplo del analizador 40 de acuerdo con una primera realización de la presente invención. Incluso aunque el procedimiento de extracción de datos de claves y datos de valores de claves a codificar usando el analizador 40 puede realizarse sobre todas las componentes (x, y, z y θ) de los datos de valores de claves, este procedimiento se describirá en los siguientes párrafos, tomando en consideración sólo una de las componentes de los datos de valores de claves (el interpolador de orientación) por conveniencia para la explicación.

Refiriéndonos a la FIG. 4A, el analizador de acuerdo con la primera realización de la presente invención incluye un dispositivo de re-muestreo 43, que muestrea una primera trayectoria de animación en base a un interpolador de orientación de entrada en un número predeterminado de secciones que tienen intervalos de una magnitud predeterminada de tiempo entre ellos y saca la trayectoria de animación muestreada al codificador de claves 200, el codificador de valores de clave 300, y el codificador de cabecera 400, el extractor de puntos de ruptura 42, que extrae un número mínimo de puntos de ruptura por lo que los errores entre la primera trayectoria de animación y la segunda trayectoria de animación generada en base a los puntos de ruptura extraídos desde la primera trayectoria de animación puede impedirse que excedan un límite de error predeterminado, y saca los puntos de ruptura extraídos al codificador de datos de claves 200, el codificador de datos de valores de claves 300, y el codificador de cabecera 400, y un selector 41, que saca el interpolador de orientación de entrada al dispositivo de re-muestreo 43 ó el extractor de puntos de ruptura 42 en respuesta a una señal de entrada externa. El extractor de puntos de ruptura 42 incluye un interpolador lineal 42a, un calculador de errores 42b, y una unidad de determinación 42c.

La FIG. 5A es un diagrama de flujo del funcionamiento del analizador 40 de acuerdo con la primera realización de la presente invención. Refiriéndonos a la FIG. 5A, el selector 41 recibe un interpolador de orientación y una señal de establecimiento desde el exterior en la etapa S325. La señal de establecimiento incluye una señal de establecimiento del procedimiento de generación usada para determinar un procedimiento de generación de los datos de claves y los datos de valores de claves a codificar y una señal de establecimiento del modo de generación para determinar un modo para la generación de los datos de claves y los datos de valores de claves a codificar.

En primer lugar se describirá el modo para la generación de los datos de claves y los datos de valores de claves en los siguientes párrafos.

El analizador 40 reduce la cantidad de datos de claves y de datos de valores de claves a codificar disminuyendo el número de fotogramas clave de una entrada del interpolador de orientación a los mismos. Se supone que el analizador 40 tiene, bien un modo de conservación de la trayectoria de animación o un modo de conservación de la clave de animación, dependiendo de la señal de establecimiento de modo introducida en el mismo desde el exterior.

En el modo de conservación de la trayectoria de animación, se usa sólo el interpolador de animación para describir la interpolación de una trayectoria de animación y el acceso aleatorio a los fotogramas clave no es necesario. Para codificar de forma efectiva un interpolador de orientación en el modo de conservación de la trayectoria de animación, los datos de claves de un interpolador de orientación existentes a lo largo de una trayectoria de animación dentro de un intervalo de error predeterminado y los datos de valores de clave correspondientes a los datos de claves pueden eliminarse.

Por otra parte, en el modo de conservación de la clave de animación es necesario tener un acceso aleatorio a los fotogramas clave usando comandos MPEG-4 BIFS, tales como 'reemplazar', 'borrar' o ´insertar´. En el modo de conservación de la clave de animación, el número de datos de claves de un interpolador de orientación no cambia. El modo de conservación de la trayectoria de animación y el modo de conservación de la clave de animación se describirán de forma más completa más adelante.

Refiriéndonos de nuevo a la FIG. 5A, el selector 41 selecciona un modo para la generación de datos de claves y los datos de valores de claves a codificar, siguiendo una entrada del modo de generación desde el exterior. En la etapa S330, el selector 41 saca el interpolador de orientación de entrada al extractor de puntos de ruptura 42, si el modo de generación de entrada es el modo de conservación de la clave de animación. Si el modo de generación de la entrada es el modo de conservación de la trayectoria de animación, el selector 41 saca el interpolador de orientación de la entrada al dispositivo de re-muestreo 43 ó el extractor de puntos de ruptura 42 junto con la información necesaria para generar los datos de claves y los datos de valores de claves en la etapa S330, en respuesta a una señal de establecimiento del procedimiento de generación introducida desde el exterior.

Específicamente, en el caso de la generación de datos de claves y de datos de valores de claves a codificar por re-muestreo, el selector 41 saca el número de los datos de claves (es decir, los intervalos de tiempo) y un modo de generación junto con el interpolador de orientación al dispositivo de re-muestreo 43. En el caso de generar los datos de claves y los datos de valores de claves a codificar extrayendo los puntos de ruptura; el selector 41 saca un error crítico entre una trayectoria de animación original y la trayectoria a generar por los puntos de ruptura extraídos y el modo de generación para el extractor de puntos de ruptura 42.

El dispositivo de re-muestreo 43 genera datos de claves muestreados y datos de valores de claves muestreados, muestreando una trayectoria de animación generada por la entrada del interpolador de orientación desde el selector 41 en intervalos de una cantidad de tiempo predeterminada, y el extractor de puntos de ruptura 42 extrae un número mínimo de puntos de ruptura por lo que puede impedirse que los errores entre la trayectoria de animación generada por el interpolador de orientación de entrada y la trayectoria de animación a generar por los puntos de ruptura extraídos exceda un límite de error predeterminado, en la etapa S335.

La FIG. 5B es un diagrama de flujo del funcionamiento del dispositivo de re-muestreo 43 de acuerdo con una realización preferida de la presente invención. Refiriéndonos a la FIG. 5B, el dispositivo de re-muestreo 43 recibe un interpolador de orientación y el número de datos de claves a re-muestrear desde el selector 41 en la etapa S502. El número (m) de datos de claves a re-muestrear puede establecerse arbitrariamente por un usuario o puede establecerse en un valor predeterminado por adelantado.

El dispositivo de re-muestreo 43 selecciona un primer punto de la trayectoria y un punto final de la trayectoria de una trayectoria de animación original generados por el interpolador de orientación de entrada y establece un valor inicial (i) de los datos de claves a re-muestrear en 1 en la etapa S504.

Después de esto, el dispositivo de re-muestreo 43 genera los datos de claves de orden i en intervalos de una cantidad predeterminada de tiempo en la etapa S506.

La FIG. 6A es un diagrama que ilustra los datos de claves originales y los datos de claves re-muestreados. Como los datos de claves del interpolador de orientación de entrada representan las localizaciones de los fotogramas clave sobre un eje temporal, los datos de claves aumentan de forma monótona pero los intervalos entre los datos de claves son irregulares, como se muestra en la FIG. 6A.

Por lo tanto, como se muestra en la FIG 6A, el dispositivo de re-muestreo 43 obtiene un intervalo de una cantidad de tiempo predeterminada dividiendo la diferencia entre los datos de claves que representan respectivamente el primer punto de la trayectoria y el punto final de la trayectoria seleccionados en la etapa S504 por el número de datos de claves a re-muestrear y a continuación re-muestrea los datos de claves a re-muestrear en intervalos de la cantidad de tiempo predeterminada.

En la etapa S508, el dispositivo de re-muestreo 43 genera los datos de valores de claves correspondientes a los datos de claves generados re-muestreando por la interpolación lineal usando la trayectoria de animación original. En otras palabras, los datos de valores de claves correspondientes a los datos de claves re-muestreados se interpolan de forma lineal usando los datos de valores de claves correspondientes a los datos de claves correctos después de los datos de claves re-muestreados y los datos de valores de claves correspondientes a los datos de claves correctos antes de los datos de claves re-muestreados.

Después de esto, en la etapa S510, el dispositivo de re-muestreo 43 verifica si el procedimiento de re-muestreo se ha realizado sobre todos los datos de claves a re-muestrear y realiza repetidamente las etapas S506 y S508 hasta que todos los datos de claves y sus datos de valores de claves correspondientes se re-muestrean.

La FIG. 5C es un diagrama de flujo de un procedimiento de extracción de puntos de ruptura de acuerdo con una primera realización de la presente invención, y las FIG. 7A hasta 7F son diagramas que ilustran cada una de las etapas de extracción de los puntos de ruptura desde un interpolador de orientación de acuerdo con una realización preferida de la presente invención.

Refiriéndonos a las FIG. 4A, 5C, y de 7A hasta 7F, el interpolador lineal 42a del extractor de puntos de ruptura 42 recibe un interpolador de orientación y un error crítico e_{th} desde el selector 41 en la etapa S520. Una trayectoria de animación constituida por el interpolador de orientación de entrada se muestra en la FIG. 7A.

El interpolador lineal 42a extrae un primer punto de la trayectoria Q_0 y un punto final de la trayectoria Q_n de la trayectoria de animación constituida por el interpolador de orientación de entrada, como se muestra en la FIG. 7A, y establece un contador (i) en 1 en la etapa S522.

El interpolador lineal 42a selecciona de forma arbitraria o secuencial los puntos de la trayectoria entre los puntos de trayectoria primero y final Q_0 y Q_n uno por uno en la etapa S524. A continuación, el interpolador lineal 42a interpola linealmente los puntos de la trayectoria que no se han seleccionado aún, usando los puntos de la trayectoria seleccionada y saca los puntos de la trayectoria seleccionada y los puntos de la trayectoria interpolados al calculador de errores 42b en la etapa S526.

El calculador de errores 42b calcula un error (e) entre la trayectoria de animación original y la trayectoria de animación candidata constituida por los puntos seleccionados de la trayectoria y los puntos interpolados de la trayectoria y saca el error (e) a la unidad de determinación 42c en la etapa S528. El procedimiento de cálculo del error (e) se describirá más adelante.

El calculador de errores 42b comprueba si entre los puntos de la trayectoria, que no se han seleccionado por el interpolador lineal 42a, existen puntos de la trayectoria, que no se han considerado cuando se calculó el error (e). Si hay puntos de la trayectoria que no se han considerado cuando se calculó el error (e), el calculador de errores 42b calcula un error entre los puntos de la trayectoria y la trayectoria de animación original en la etapa S530 realizando repetidamente las etapas S524 hasta S528.

La FIG. 7C es un diagrama que ilustra las etapas S524 hasta S530. Refiriéndonos a la FIG. 7C, el interpolador lineal 42a extrae un punto de ruptura Q_1 correspondiente a los datos de claves en un momento predeterminado de tiempo k_1 y genera una primera trayectoria de animación candidata por interpolación lineal de los puntos de la trayectoria entre el primer punto de la trayectoria Q_0 y el punto de ruptura Q_1. El calculador de errores 42b calcula un error e_1 entre la trayectoria de animación original y la primera trayectoria de animación candidata. Después de esto, del mismo modo, el interpolador lineal 42a selecciona otro punto de ruptura Q_k y genera una trayectoria de animación candidata de orden k por la interpolación lineal de los puntos de la trayectoria entre el primer punto de la trayectoria Q_0 y el punto de ruptura Q_k y entre el punto de ruptura Q_k y el punto final de la trayectoria Q_n. El calculador de errores 42b calcula un error (e_k) entre la trayectoria de animación original y la trayectoria de animación candidata de orden k.

Si las etapas S524 hasta S530 se han realizado sobre todos los puntos de la trayectoria que no se han seleccionado por el interpolador lineal 42a, los errores entre la trayectoria de animación original y las trayectorias de animación candidatas generadas cada una siguiendo las etapas S524 hasta S530 se sacan a la unidad de determinación 42c. A continuación, la unidad de determinación 42c selecciona un punto de ruptura, que forma una trayectoria de animación candidata que tiene el menor error con la trayectoria de animación original, y aumenta el valor del contador (i) en 1 en la etapa S532.

La unidad de determinación 42c comprueba si un error (e) entre la trayectoria de animación original y la trayectoria de animación candidata constituida por los puntos de ruptura extraídos es mayor que el error crítico e_{th} y el valor del contador (i) es mayor que el número (n) de datos de claves, es decir, el número de puntos de trayectorias entre el primer punto de trayectoria Q_0 y el punto final de la trayectoria Q_n, en la etapa S534.

Si el error (e) es más pequeño que el error crítico e_{th}, significa que se han extraído todos los puntos de ruptura requeridos para la codificación. Si el número de puntos de ruptura finalmente seleccionados como los que se van a codificar, es igual a 'n', significa que todos los puntos de la trayectoria del procedimiento de extracción de los puntos de ruptura están completos.

Sin embargo, si el número de puntos de ruptura extraídos es menor que n y el error (e) es mayor que el error crítico e_{th}, lo que significa que aún existen puntos de ruptura a extraer, los puntos de ruptura seleccionados se sacan al interpolador lineal 42a, y a continuación se realizan de nuevo las etapas desde S524 hasta S532.

En lo sucesivo, se supone que los datos a sacar desde el dispositivo de re-muestreo 43 y el extractor de puntos de ruptura 42 al codificador de datos de claves 200 y el codificador de datos de valores de claves 300 cuando el modo de generación es un modo de conservación de la trayectoria de animación se describirán en los siguientes párrafos.

El dispositivo de re-muestreo 43 saca los datos de claves muestreados y los datos de valores de claves muestreados al codificador de datos de claves 200 y el codificador de datos de valores de claves 300, respectivamente, como los datos de claves y los datos de valores de claves a codificar respectivamente.

En lo sucesivo, se describirán los datos de claves y los datos de valores de claves sacados procedentes del extractor de puntos de ruptura 42 dependiendo del modo de generación con referencia a la FIG. 8.

Como se muestra en la FIG. 8, suponiendo finalmente que los puntos de ruptura extraídos se denominan como 0, 3, 6 y 8, los datos de claves y los datos de valores de claves correspondientes a los puntos de ruptura 0, 3, 6 y 8 se sacan con un indicador de selección de clave, que se muestra en la siguiente tabla

Tabla 2

Datos de Claves de la Trayectoria Original	P0	P1	P2	P3	P4	P5	P6	P7	P8
Indicador de Selección de Clave	1	0	0	1	0	0	1	0	1

La estructura del analizador 40 de acuerdo con la primera realización de la presente invención se ha descrito anteriormente. Sin embargo, el analizador 40 puede estar constituido sólo por el extractor de puntos de ruptura 42 sin el selector 41 y el dispositivo de re-muestreo 43 o puede estar constituido sólo por el dispositivo de re-muestreo 43 sin el selector 41 y el extractor de puntos de ruptura 42, lo que es obvio para los expertos en la materia.

En lo sucesivo, se describirá otro ejemplo del analizador 40 de acuerdo con una segunda realización de la presente invención.

Refiriéndonos a la FIG. 4B, el analizador 40 de acuerdo con la segunda realización de la presente invención incluye un dispositivo de re-muestreo 45, que recibe y re-muestrea un interpolador de orientación, y un extractor de puntos de ruptura 46, que extrae los puntos de ruptura del interpolador de orientación re-muestreado y saca los datos de claves y los datos de valores de claves a codificar. El extractor de puntos de ruptura 46 en la segunda realización de la presente invención, como en la primera realización de la presente invención, también incluye un interpolador lineal 46a, un calculador de errores 46b y una unidad de determinación 46c.

Cuando se introduce un interpolador de orientación en el analizador 40, el dispositivo de re-muestreo 45 re-muestrea una primera trayectoria de animación constituida por el interpolador de orientación en un número de secciones predeterminado que tienen un intervalo de una cantidad de tiempo predeterminada entre sí.

El dispositivo de re-muestreo 45 saca el interpolador de orientación que consiste de datos de claves muestreados y datos de valores de claves muestreados para el interpolador lineal 46a del extractor de puntos de ruptura 46.

El interpolador lineal 46a interpola un interpolador de orientación realizando las etapas S522 hasta S526 mostradas en la FIG. 5C y saca el interpolador de orientación interpolado al calculador de errores 46b. El calculador de errores 46b calcula un error entre la primera trayectoria de animación y la segunda trayectoria de animación constituida por el interpolador de orientación interpolado realizando las etapas S528 y S530. La unidad de determinación 46c selecciona un punto de la trayectoria que conducirá a un error mínimo entre la primera y la segunda trayectorias de animación, verifica si el error correspondiente es mayor que un error crítico e_{th} y si se han seleccionado todos los puntos de la trayectoria de la primera trayectoria de animación, y genera los datos de claves y los datos de valores de claves a codificar.

Como se ha descrito anteriormente, en el analizador 40 de acuerdo con la segunda realización de la presente invención, la operación del dispositivo de re-muestreo 45 y el extractor de puntos de ruptura 46 es lo mismo que la operación de los elementos correspondientes en la primera realización de la presente invención excepto que el extractor de puntos de ruptura 46 recibe un interpolador de orientación consistente de la salida de los datos de claves y los datos de valores de claves desde el dispositivo de re-muestreo 45 y el procedimiento de extracción de puntos de ruptura se realiza sobre la trayectoria de animación constituida por la entrada del interpolador de orientación desde el dispositivo de re-muestreo 45.

En lo sucesivo, se describirá un ejemplo del analizador 40 de acuerdo con una tercera realización de la presente invención con referencia a la FIG. 4C.

Refiriéndonos a la FIG. 4C, el analizador 40 incluye un extractor de puntos de ruptura 48, que recibe un interpolador de orientación, extrae los puntos de ruptura desde una primera trayectoria de animación constituida por el interpolador de orientación, y saca los datos de claves y los datos de valores de claves, y un dispositivo de re-muestreo 49, que re-muestrea una segunda trayectoria de animación constituida por un interpolador de orientación

consistente de la entrada de datos de claves y datos de valores de claves desde el extractor de puntos de ruptura 48 en intervalos de una cantidad de tiempo predeterminada. El extractor de puntos de ruptura 48, como los de la primera y segunda realizaciones de la presente invención, también incluye un interpolador lineal 48a, un calculador de errores 48b y una unidad de determinación 48c.

El extractor de puntos de ruptura 48, como el de la primera realización de la presente invención, saca los datos de claves y los datos de valores de claves extraídos desde la primera trayectoria de animación al dispositivo de re-muestreo 49.

El dispositivo de re-muestreo 49 re-muestrea una trayectoria de animación constituida por un interpolador de orientación consistente de la entrada de datos de claves y datos de valores de claves desde el extractor de puntos de ruptura 48 en intervalos de una cantidad de tiempo predeterminada y saca los datos de claves y los datos de valores de claves a codificar. La función del dispositivo de re-muestreo 49 es la misma que en la primera y segunda realizaciones de la presente invención, y de este modo no se repetirá su descripción en este punto.

La salida de los datos de clave y los datos de valores de claves desde el analizador 40 en las realizaciones de la primera hasta la tercera de la presente invención se sacan al codificador de datos de claves 200 y al codificador de datos de valores de claves 300, respectivamente.

En lo sucesivo, se describirán la estructura y el funcionamiento del codificador de los datos de claves 200 con referencia las FIG. 9A hasta 12J,

La FIG. 9A es un diagrama de bloques de un codificador de datos de claves de acuerdo con una realización preferida de la presente invención. Refiriéndonos a la FIG. 9A, un codificador de datos de clave 200 incluye un codificador de claves lineal 900, un dispositivo de cuantización 910, un procesador de DPCM 920, un dispositivo de desplazamiento 930, un procesador de plegado 940, un procesador de 950 DND, y un codificador de entropía 960.

El codificador de claves lineal 900 identifica una región donde los datos de clave aumentan linealmente en un intervalo entero de datos de claves y codifica la región. El dispositivo de cuantización 910 cuantiza la entrada de datos de clave dentro del mismo usando un procedimiento de cuantización capaz de minimizar un error de cuantización. El procesador de DPCM 920 recibe los datos de claves cuantizados y genera los datos diferenciales de los datos de claves. El dispositivo de desplazamiento 930 resta un dato diferencial que tiene la frecuencia más alta de entre todos los datos diferenciales a partir de los datos diferenciales. El procesador de plegado 940 transfiere todos los datos diferenciales bien a una región de números positivos o a una región de números negativos. El procesador de DND 950 reduce el intervalo de datos diferenciales de los datos de claves realizando una operación de división y a continuación realizando de forma selectiva una operación de división de subida o una operación de división de bajada. El codificador de entropía 960 codifica los datos diferenciales usando una función de SignedAAC o UnsignedAAC sobre cada uno de los planos de bits.

En lo sucesivo, se describirá la operación del codificador de los datos de claves 200 más completamente con referencia a la FIG. 10A. La FIG. 10A es un diagrama de flujo de un procedimiento de codificación de datos de claves de acuerdo con una realización preferida de la presente invención. Cuando los datos de claves se introducen dentro de un aparato para la codificación de un interpolador de orientación, información, tal como el número de datos de claves y el número de dígitos de cada uno de los datos de claves, se introduce en el codificador de cabeceras 40 y se codifica. El codificador de claves lineal 900 busca una región en los datos de claves de entrada donde existen tramas de claves en ciertos intervalos temporales, los datos de claves tienen la misma diferencia, y los datos de claves cambian linealmente, y la región lineal buscada se codifica en primer lugar en la etapa S9000.

Software de aplicación de 3D famosos, tales como 3DMax o Maya, generan animación basada en tramas de claves que usan claves que tienen un intervalo de tiempo predeterminado entre las mismas en regiones específicas. En este caso, es posible codificar fácilmente datos de claves usando los datos de claves del comienzo y del final de una región de datos de claves lineal y el número de tramas de claves existentes entre ellas. Por consiguiente, la predicción lineal es muy útil para la codificación de claves en ciertas regiones que usan un interpolador.

Se usa la siguiente ecuación para la predicción lineal.

$$t(i) = \frac{t_E - t_S}{E - S} + t_S \quad (0 \le i \le E - S,\ S < E) \quad \cdots(4)$$

En este punto, t_S representa los datos de una clave donde comienza la región parcialmente lineal, t_E representa los datos de una clave donde termina la región parcialmente lineal, S representa un índice de t_S, y E representa un índice de t_E. El error entre los datos de claves reales en una región específica que varía desde los datos de clave de orden S a los datos de claves de orden E y los datos de clave predichos de forma lineal que siguen la ecuación (4) pueden calcularse usando la siguiente ecuación.

$$e_i = t(i) - t_{i+S} = + t_S - t_{i+S} \quad \frac{t_E - t_S}{E - S} i + t_S - t_{i+S} \quad \ldots(5)$$

Si el valor máximo entre los errores calculados usando la ecuación (5) no es mayor de un límite de error predeterminado, t_i puede considerarse co-lineal en la región [t_S, t_E] o dentro de un cierto intervalo de errores. Si el valor del error máximo t_i es, o no, co-lineal con la región específica se determina usando la siguiente ecuación (6).

$$E_p = \underset{i=0,\ldots(E-S)}{MAX} |e_i| = \underset{i=0,\ldots(E-S)}{MAX} \left| \frac{t_E - t_S}{E - S} i + t_S - t_{i+S} \right| \quad \ldots(5)$$

Si $E_p \leq \frac{1}{2^{nBits+1}}$, t_i es co-lineal con la región [t_S, t_E]. En este punto, nBits representa el número de bits usados para la codificación.

Si el codificador de claves lineal 900 busca la región parcialmente lineal, los datos de claves de comienzo y de final de la región de datos de claves parcialmente lineal se sacan al convertidor de números en punto flotante 905. El número de claves incluidas en la región de datos de claves lineal se saca al codificador de cabecera 400 y se codifica. Es posible reducir considerablemente la cantidad de datos a codificar usando la codificación lineal.

Los datos de claves de comienzo y los datos de claves de final se codifican usando una transformación de números en punto flotante, que se describirá más adelante.

El convertidor de números de punto flotante 905 convierte los datos de claves representados en el sistema binario al sistema decimal para codificar los datos de claves de comienzo y los datos de claves de final.

Un ordenador almacena los números de punto flotante como números binarios de 32 bits. Si se da un número de punto flotante representado en el sistema binario, el convertidor de números de punto flotante 905 convierte el número de punto flotante en una mantisa y un exponente en el sistema decimal, y este procedimiento se expresa por la siguiente ecuación.

$$\underbrace{\text{mantisa\_binaria} * 2^{\text{exponente\_binario}}}_{\text{número en punto\_flotante en sistema binario}} = \underbrace{\text{mantisa} * 10^{\text{exponente}}}_{\text{número en punto\_flotante en sistema decimal}} \quad \ldots(7)$$

Por ejemplo, un número en punto flotante 12,34 puede convertirse por un ordenador en el número binario, que se muestra en lo siguiente.

0 10001010111000010100011 10000010
1 2 3

1: el signo
2: la mantisa en el sistema binario
3: el exponente en el sistema binario

El número binario puede convertirse en un número decimal siguiendo la ecuación (7), que se muestra en lo siguiente

0 1234 2
1 2 3

1: el signo
2: la mantisa en el sistema decimal
3: el exponente en el sistema decimal

Para incluir una mantisa y un exponente en el sistema decimal en un flujo de bits, debe calcularse el número de bits requerido para representar la mantisa y el exponente. El exponente tiene un valor entre -38 y +38 y de este modo puede expresarse junto con su signo usando 7 bits. El número de bits requerido para representar la mantisa depende del número de dígitos. Los valores de la mantisa y el número de bits requerido para representar la mantisa se muestran en la siguiente tabla.

Tabla 4

Valores de mantisa	Dígitos de mantisa	Número de bits requerido
0	0	0
1 – 9	1	4
10 – 99	2	7
100 – 999	3	10
1000 – 9999	4	14
10000 – 99999	5	17
100000 – 999999	6	20
1000000 – 9999999	7	24

Los datos de claves de comienzo y de final de la región lineal de datos de claves que se ha buscado y se ha convertido usando los procedimientos mencionados anteriormente, se codifican siguiendo el procedimiento de codificación mostrado en la FIG. 10B, se sacan al codificador de cabeceras 400, y se almacenan en el flujo de bits.

La FIG. 10B muestra un procedimiento de codificación de dos números de entrada de punto flotante realizado en el convertidor de números de punto flotante 905. El modo en el que el convertidor de números de punto flotante 905 codifica un número de punto flotante se describirá con referencia a la FIG. 10B.

El convertidor de números de punto flotante 905 recibe el número de dígitos Kd de los datos de claves originales, los datos de claves de comienzo S, y los datos de claves de final E y los convierte en la etapa S9040 siguiendo la ecuación (7).

El convertidor de números en punto flotante 905 codifica en primer lugar S. En particular, el convertidor de números en punto flotante 905 comprueba si el número de dígitos de S es o no, diferente de Kd. Si el número de dígitos de S es diferente de Kd, se obtiene el número de dígitos de S, y se saca al codificador de cabeceras 400 en la etapa S9042. El convertidor de números en punto flotante 905 obtiene el número de dígitos de S usando la función Digit ().

Si el número de dígitos de S es mayor que 7, se saca S al codificador de cabeceras 400 usando un número predeterminado de bits (en la presente invención, se usan 32 bits siguiendo el modo de los números de punto flotante de la Normativa de IEEE 754) en la etapa S9043 de modo que el número de dígitos de S puede incluirse en el flujo de bits.

Si el número de dígitos de S no es 0 y es menor que 7, el convertidor de números de punto flotante 905 saca el signo de S al codificador de cabecera 400 en la etapa S9044. El número de bits requerido para codificar el valor absoluto de la mantisa de S, se obtiene usando la Tabla 4. A continuación, el valor absoluto de la mantisa de S se saca al codificador de cabecera 400 usando el número de bits obtenido usando la Tabla 4, en la etapa S9045. El convertidor de números de punto flotante 905 calcula el exponente de S, saca el signo de S al codificador de cabecera 400, y saca el exponente al codificador de cabecera 400 como un número predeterminado de bits, por ejemplo, 6 bits, en la etapa S9046. Tal transformación de datos de claves hace posible reducir considerablemente el número de bits incluidos en el flujo de bits.

Si el número de dígitos de S es 0, la codificación de los datos de claves de comienzo termina y el procedimiento va a la etapa de transformación de los datos de claves de final E porque cuando el número de dígitos de S es 0, el número en punto flotante es también 0 que no requiere codificación.

Después de la transformación y la codificación de los datos de claves de comienzo S, el convertidor de números de punto flotante 905 convierte los datos de claves de final E. La transformación de E es casi la misma que la de S. En particular, el convertidor de números de punto flotante 905 comprueba si el exponente de E es o no el mismo que el de S en la etapa S9047. Si el exponente de E es el mismo que el de S, se saca sólo un bit indicador que representa que el exponente de E es el mismo que el exponente de S al codificador de cabecera 400. Si el exponente de E no es el mismo que el de S, el exponente de E así como el bit indicador se sacan al codificador de cabecera 400 del mismo modo que se ha sacado el exponente de S al codificador de cabecera 400, en la etapa S9048.

Los datos de claves de entre los datos de claves de entrada que no pertenecen a la región lineal se introducen en el dispositivo de cuantización 910 y se realiza la cuantización de acuerdo con un tamaño de bits de cuantización predeterminado, es decir, nKeyQbit.

Sin embargo, en el caso de una decodificación de los datos de claves cuantizados usando un decodificador, es imposible recuperar perfectamente los datos de claves originales debido a los errores entre los datos de claves originales y los datos de claves cuantizados. Por lo tanto, el dispositivo de cuantización 910 de la presente invención

obtiene un valor máximo y un valor mínimo de entre los datos de claves de entrada y realiza la cuantización de los datos de claves de entrada usando los valores máximo y mínimo. Además, la presente invención incluye un minimizador de errores de cuantización 915 de modo que el error entre los datos de claves originales y sus datos de claves cuantizados puede minimizarse usando los valores máximo y mínimo de entre los datos de claves de entrada.

El minimizador de errores de cuantización 915 realiza la cuantización o la cuantización inversa de los datos de claves de entrada con anticipación usando un procedimiento para controlar un intervalo de cuantización de modo que el error de cuantización puede minimizarse, en la etapa S9100.

En particular, si el valor máximo de mezcla usado para la cuantización se representa por Max, el valor mínimo a controlar para la cuantización se representa por Min, el valor de entrada se representa por X_i, y el número de bits usado para la cuantización se representa por nQuantBit, entonces, se obtienen un valor de entrada cuantizado $\tilde{X}_i$, un valor de la cuantización inversa $\hat{N}_i$

y un error e_i usando la siguiente ecuación.

$$\tilde{X}_i = floor(\frac{X_i - Min}{Max - Min} * (2^{nQuantBit} - 1) + 0,5) \quad \ldots(8)$$

$$\hat{X}_i = \frac{\tilde{X}_i * (Max - Min)}{2^{nQuantBit} - 1} + Min$$

$$e_i = X_i - \hat{X}_i$$

Hay dos procedimientos para reducir la suma de para reducir la suma $\sum e_i$ de los errores. Uno es un procedimiento para reducir la suma de los errores controlando continuamente Min hasta que la suma de los errores se minimiza. El otro es como sigue.

En primer lugar, asumamos que $X_i \equiv (i + n)\,\Delta x + \varepsilon_i$ donde X_i indica una secuencia de datos de claves de entrada, Δx indica un tamaño de paso básico de los datos de entrada, n es un número entero arbitrario, y ε_i indica un ruido aleatorio de media cero.

A continuación, cuando $d_i = X_i - X_{i-1} = \Delta x + (\varepsilon_i - \varepsilon_{i-1})$, $\Delta x = E[d_i]$ y $Min = Max - \Delta' x * (2^{nQuantbit} - 1)$.

Min, que puede hacer posible minimizar un error de cuantización, y Max se introducen al dispositivo de cuantización 910 y se usan para la cuantización de los datos de claves.

El dispositivo de cuantización 910 recibe los valores máximo y mínimo Max y Min que puede minimizar un error de cuantización y cuantiza los datos de claves $fKey_i$ en la etapa S9200, siguiendo la ecuación (9).

$$nQKey_i = floor(\frac{fKey_i - fKeyMin}{fKeyMax - fKeyMin}(2^{nKeyQBit} - 1) + 0.5) \quad \ldots(9)$$

En este punto, i indica un índice de datos de claves cuantizados, $nQKey_i$ indica una disposición de números enteros de los datos de claves cuantizados, $fKey_i$ indica una disposición de números de punto flotante de los datos de claves cuantizados, fKeyMax indica una entrada de valor máximo desde el minimizador de errores de cuantización 915, fKeyMin indica una entrada de valor mínimo desde el minimizador de errores de cuantización 915, y nKeyQBit indica un tamaño de bits de cuantización. En la ecuación (9), la función floor (v) es una función que saca un número entero máximo no mayor que un cierto valor en punto flotante v.

El dispositivo de cuantización 910 de la presente invención puede que no use tal algoritmo para la reducción de un error de cuantización, en cuyo caso la cuantización se realiza simplemente usando los valores máximo y mínimo fKeyMax y fKeyMin de entre los datos de claves de entrada.

Un procedimiento de cuantización de la presente invención se describirá más completamente con referencia a la FIG. 10C.

El dispositivo de cuantización 910 recibe datos de claves en la etapa S9210 y comprueba si los valores máximo y mínimo MAX y MIN se introducen desde el minimizador de errores de cuantización 915 en la etapa S9220.

Si se introducen MAX y MIN, el dispositivo de cuantización 910 fija los valores máximo y mínimo fKeyMax y fKeyMin para la cuantización como MAX y MIN, respectivamente, en la etapa 3230 saca los valores máximo y mínimo fijados de nuevo fKeyMax y fKeyMin al convertidor de números de punto flotante 905, Los valores máximo y mínimo fKeyMax y fKeyMin se convierten y se codifican a través del procedimiento de transformación de números de punto flotante mencionado anteriormente y se sacan al codificador de cabeceras 400 de modo que pueden incluirse en una cabecera de claves para su uso en la decodificación.

Si no hay ninguna entrada de valores desde el minimizador de errores de cuantización 915, el dispositivo de cuantización 910 fija los primeros datos de claves $fKey_0$ y los datos de claves finales $fKey_{N-1}$ como el valor mínimo fKeyMin y el valor máximo fKeyMax respectivamente, en la etapa S9240.

A continuación, el dispositivo de cuantización 910 comprueba si el valor máximo fKeyMax es o no menor que 1 pero mayor que 0 y si el valor mínimo fKeyMin es o no mayor que 0 en la etapa S9250. Si el valor máximo fKeyMax no es menor que 1 o no es mayor que 0, los valores máximo y mínimo fKeyMax y fkeyMin se sacan al convertidor de números de punto flotante 905 y se convierten y se codifican a través de la transformación de números de punto flotante mencionada anteriormente. A continuación, los valores máximo y mínimo fKeyMax y fKeyMin que se han convertido y codificado se incluyen en la cabecera de claves en la etapa S9260 de modo que pueden usarse en la decodificación.

Por otra parte, si el valor máximo fKeyMax es menor que 1 y el valor mínimo fKeyMin es mayor que 0, se incluirá un indicador que indica si los valores máximo y mínimo fKeyMax y fKeyMin se incluirán en la cabecera de claves para usarse en la decodificación que se comprueba en la etapa S9270. Si se establece el indicador de modo que los valores máximo y mínimo fKeyMax y fKeyMin pueden incluirse en la cabecera de claves, se realiza la etapa S9260 de modo que los valores máximo y mínimo fKeyMax y fKeyMin se sacan al codificador de cabeceras 400. Si el indicador no se establece, el dispositivo de cuantización 910 no permite que los valores máximo y mínimo fKeyMax y fKeyMin se incluyan en la cabecera de claves.

En un caso en el que los valores máximo y mínimo fKeyMax y fKeyMin no se incluyen en la cabecera de claves, el codificador de datos de claves y el decodificador de datos de claves se supone que realizan la codificación y la decodificación, respectivamente, estableciendo los valores máximo y mínimo fKeyMax y fKeyMin a 1 y a 0, respectivamente. En este caso, el dispositivo de cuantización 910 establece los valores máximo y mínimo fKeyMax y fKeyMin a 1 y a 0, respectivamente en la etapa S9280. Los valores máximo y mínimo fKeyMax y fKeyMin ya se conocen para el decodificador de datos de claves de modo que no necesitan incluirse en la cabecera de claves.

El dispositivo de cuantización 910 cuantiza los datos de claves de entrada sustituyendo los valores máximo y mínimo fKeyMax y fKeyMin que se han establecido a través del procedimiento mencionado anteriormente en la ecuación (9) y saca los datos de claves cuantizados al procesador de DPCM 920 en la etapa S9290.

El procesador de DPCM 920 recibe los datos de claves cuantizados y realiza la DPCM sobre los datos de claves cuantizados un número predeterminado de veces. A continuación el procesador de DPCM 920 saca el orden de la DPCM, por el cual puede obtenerse un valor mínimo en el grado de dispersión, y los datos intra claves obtenidos en cada uno de los ciclos de DPCM, al codificador de cabecera 400. El procesador de DPCM 920 saca los datos diferenciales generados por la DPCM al dispositivo de desplazamiento 930 en la etapa S9300.

Refiriéndonos a la FIG. 10D, el procesador de DPCM 920 realiza la DPCM sobre los datos de claves de entrada un número predeterminado de veces y almacena el número de ciclos de DPCM como el orden de DPCM en la etapa S9310. En una realización preferida de la presente invención, la DPCM puede realizarse tres veces.

Después de esto, el procesador de DPCM 920 calcula el grado de dispersión de los resultados de cada uno de los ciclos de DPCM en la etapa S9320. En este punto, el grado de dispersión puede representarse por la dispersión, la desviación estándar, o la desviación en cuartiles, y en una realización preferida de la presente invención, puede usarse la desviación en cuartiles.

A continuación, el procesador de DPCM 920 selecciona un ciclo de DPCM por el cual puede obtenerse un valor mínimo en el grado de dispersión y saca los resultados del orden seleccionado de DPCM al dispositivo de desplazamiento 930. El ciclo seleccionado de DPCM, los datos de intra claves de cada uno de los ciclos de DPCM y otros elementos de información requeridos para la DPCM se sacan al codificador de cabecera 400 en la etapa S9330. Sin embargo, en una realización preferida de la presente invención, la DPCM sólo se realiza una vez si el número de claves es menor de 5. Por ejemplo, se realiza un primer ciclo de DPCM siguiendo la ecuación (10).

$$\Delta_i = nQKey_{i+1} - nQKey_i \quad \ldots(10)$$

En este punto, i indica un índice de los datos de claves cuantizados, nQKey, indica una disposición de números enteros, y Δ_i indica los datos diferenciales.

El procesador de DPCM 920 calcula el número de bits requerido para codificar los resultados del ciclo seleccionado de DPCM y los datos diferenciales de los datos de claves que se han generado por la DPCM en una memoria predeterminada (nQsep_DPCM) en la etapa S9340. El cálculo del número de bits requerido para la codificación también puede realizarse más tarde en una etapa posterior de selección de los datos de claves a codificar, hecho que es obvio para los expertos en la materia.

El dispositivo de desplazamiento 930 selecciona un dato diferencial (en adelante en este documento, denominado como un modo) que tiene la frecuencia más alta de entre los datos diferenciales introducidos desde el procesador de DPCM 920. A continuación, el dispositivo de desplazamiento 930 resta el modo de todos los datos diferenciales en la etapa S9400 de modo que la mayor parte de los datos a codificar se disponen alrededor de 0 y el número de bits requerido para la codificación puede disminuirse.

Tal operación de desplazamiento se realiza restando el modo nKeyShift de todos los datos de claves cuantizados, lo que se expresa por la siguiente ecuación.

$$shift\ (nQKey_i) = nQKey_i - nKeyShift \qquad \ldots(11)$$

En este punto, i indica un índice de los datos de claves cuantizados, $nQKey_i$ indica una disposición de números enteros, y nKeyShift indica un valor de modo. Como resultado de la operación de desplazamiento, los datos diferenciales que tienen la frecuencia más alta se hacen 0 de modo que el número de bits requerido para la codificación puede reducirse considerablemente.

Los datos de claves que han pasado a través de la operación de desplazamiento se sacan al procesador de plegado 940 y el procesador de DND 950, y el valor de modo de nKeyShift se saca al codificador de cabecera 400 de modo que se incluye en la cabecera de claves.

El procesador de plegado 940 realiza una operación de plegado sobre las salidas del dispositivo de desplazamiento 930 y saca los resultados de la operación de plegado al procesador de DND 950 en la etapa S9500.

La operación de plegado se usa para reducir el intervalo de datos diferenciales que están ampliamente dispersos tanto sobre una región de números positivos como una región de números negativos concentrándolos bien en una región de números positivos o negativos. En la presente realización la operación de plegado se realiza siguiendo la ecuación (12) para concentrar los datos diferenciales en la región de números positivos.

$$fold\ (nQKey_i) = 2 \cdot nQKey_i \qquad (si\ nQKey_i \geq 0) \qquad \ldots(12)$$

$$= 2 \left| nQKey_i \right| - 1 \qquad (si\ nQKey_i < 0)$$

En este punto, i indica un índice de los datos de claves cuantizados, $nQKey_i$ indica una disposición de números enteros. Como resultado de la operación de plegado, los datos diferenciales positivos se convierten en números pares y los datos diferentes negativos se convierten en números impares.

El procesador de plegado 940 calcula el número de bits requerido para codificar los datos diferenciales que han pasado a través de la operación de plegado y los almacena en una memoria predeterminada nQStep_fold. En esta etapa, el cálculo del número de bits requeridos para la codificación puede realzarse más adelante en una etapa posterior de selección de los datos diferenciales a codificar en entropía, lo que es obvio para los expertos en la materia. Los datos generados por la operación de plegado en el procesador de plegado 940 se sacan al procesador de DND 950.

Para mejorar la eficacia de la codificación de entropía, el procesador de DND 950 realiza una operación de DND sobre los datos diferenciales de entrada de los datos de claves un número predeterminado de veces, reduciendo de este modo el intervalo de los datos diferenciales en la etapa S9600.

Refiriéndonos a la FIG. 9B, el procesador de DND 950 incluye un operador de DND 952, que realiza una operación de DND de los datos diferenciales, un primer selector de datos diferenciales 954, que selecciona los datos diferenciales a codificar en entropía en base al número de bits a codificar, un operador de desplazamiento hacia arriba (shift – up) 956, que realiza una operación de desplazamiento hacia arriba sobre los datos diferenciales que han pasado a través de la operación de DND, y un segundo selector de datos diferenciales 958, que selecciona de entre los datos diferenciales que han pasado sólo a través de la operación de DND y los datos diferenciales que han pasado a través de la operación de desplazamiento hacia arriba los que tienen un grado de dispersión menor y saca los datos diferenciales seleccionados al codificador de entropía 960.

La operación de DND realizada en el operador de DND 952 se describirá en lo siguiente.

Cuando los datos diferenciales que han pasado a través de la operación de plegado en el procesador de plegado 940 se introducen al operador de DND 952, se dividen en dos grupos, y el grupo de los datos diferenciales que tiene un mayor intervalo que el otro grupo de datos diferenciales se mueve a la región de números positivos por una

función de división. La función de división se define por la siguiente ecuación.

$$\textbf{división}\ (nQKey_j,\ nKeyMax) \qquad \ldots(13)$$

$$= nQKey_j - (nKeyMax + 1) \qquad (si\ nQKey_j > nKeyMax / 2)$$

$$= nQKey_j \qquad (si\ nQKey_j \leq nKeyMax / 2)$$

En este punto, j indica un índice de la entrada de datos diferenciales, $nQKey_j$ indica una disposición de números enteros, y nKeyMax indica un valor máximo de entre los datos diferenciales que han pasado a través de la operación de plegado. Especialmente, en el caso en el que la mayor parte de los datos diferenciales están densamente poblados a lo largo de fronteras de toda la región representada por todos los datos diferenciales, es posible reducir considerablemente toda la región de todos los datos diferenciales usando la operación de división.

Después de la operación de división, se calcula el grado de dispersión, en cuyo caso el tamaño de bits requerido para la codificación se usa como una medida del grado de dispersión de modo que puede seleccionarse un valor mínimo en el tamaño de bits para la codificación.

Después de la operación de DND, se realiza además una diferente clase de operación de DND, es decir una operación de división hacia arriba o una operación de división hacia abajo. Se determina además si se realizará una operación de división hacia arriba o una operación de división hacia abajo dependiendo de ambos el tamaño del intervalo positivo de los da tos diferenciales y el tamaño del intervalo negativo de los datos diferenciales.

Si en intervalo de los datos diferenciales que tienen valores positivos es mayor que el intervalo de los datos diferenciales que tienen valores negativos, se realiza una operación de división hacia abajo definida por la siguiente ecuación.

$$\textbf{división hacia abajo}\ (nQKey_j,\ nKeyMax) \qquad \ldots(14)$$

$$= -2\ (nKeyMax - nQKey_j + 1) + 1 \qquad (si\ nQKey_j > nKeyMax / 2)$$

$$= nQKey_j \qquad (si\ 0 \leq nQKey_j \leq nkeyMax / 2)$$

$$= 2.\ nQKey_j \qquad (si\ nQKey_j < 0)$$

Por el contrario, si el intervalo de los datos diferenciales que tienen valores positivos es mayor que el intervalo de los datos diferenciales que tienen valores negativos, se realiza una operación de división hacia arriba definida por la siguiente ecuación.

$$\textbf{división hacia arriba}\ (nQKey_j,\ nKeyMin) \qquad \ldots(15)$$

$$= nQKey_j \qquad (nQKey_j \geq 0)$$

$$= 2.\ nQKey_j \qquad (nKeyMin / 2 \leq nQKey_j \leq 0)$$

$$= 2\ (nKeyMin - nQKey_j - 1) + 1 \qquad (nQKey_j < nKeyMin / 2)$$

En las ecuaciones (14 y (15), j indica un índice de los datos de claves cuantizados, $nQKey_j$ representa una disposición de números enteros, nKeyMax indica un valor máximo de $nQKey_j$, y nKeyMin indica un valor mínimo de $nQKey_j$.

El funcionamiento del operador de DND 952 se describirá a continuación con referencia a la FIG. 10E.

Cuando los datos diferenciales de los datos de claves de entrada se introducen desde el procesador de plegado 940, el operador de DND 952 obtiene el valor máximo de nKeyMax y el valor mínimo de nKeyMin de entre los datos diferenciales de entrada en la etapa S9610. A continuación, el operador de DND 952 compara el valor absoluto de nKeyMax con el de nKeyMin en la etapa S9620. Si nKeyMax no es menor que el valor absoluto de nKeyMin, el operador de DND 952 fija nKeyMax como un valor máximo en el ciclo actual del funcionamiento de DND en la etapa S9622.

El operador de DND 952 comprueba si el orden de las operaciones de DND es 1, en otras palabras, si el orden de la operación de DND es 1, en la etapa S9624, y si es así, el operador de DND 952 realiza una operación de división sobre los datos diferenciales de entrada en la etapa S9630 sustituyendo el valor máximo nKeyMax en la ecuación (13).

Después de esto, el operador de DND 952 mide el tamaño de bits requerido para la codificación del intervalo de

datos diferenciales que se ha reducido usando la operación de división, en la etapa S9640, usando la función getQBit (). Si el orden de la operación de DND resulta ser 1 en la etapa S9650, el tamaño de bits requerido para la codificación se almacena como un valor nQBitDND que indica el tamaño mínimo de bits para la codificación, y el orden de la operación de DND se aumenta en 1 en la etapa S9655.

A continuación, el procesador de DND 952 realiza las etapas S9610 hasta S9622 de nuevo. Si el orden de la operación de DND no es 1 en la etapa S9624, el operador de DND 952 realiza una operación de división hacia abajo en la etapa S9634 sustituyendo el valor máximo nKeyMax en la ecuación (14). El operador de DND 952 calcula el número de bits requeridos para la codificación de los datos diferenciales que han pasado a través de la operación de división hacia abajo, en la etapa S9640. Si el número es más pequeño que el valor mínimo nQBitDND almacenado en el ciclo anterior de la operación de DND, reemplaza el tamaño mínimo de bits requerido para la codificación después de la operación de DND en la etapa S9658.

Si el valor absoluto del valor mínimo nKeyMin parece que es mayor que el del valor máximo nKeyMax en la etapa S9620, el valor máximo en el ciclo actual de la operación de DND se renueva como un valor mínimo en la etapa S9623, y a continuación se realiza una operación de división hacia arriba en la etapa S9628 sustituyendo el valor mínimo para nKeyMin en la ecuación 15. Después de esto, el operador de DND 952 calcula el número de bits para la codificación de los datos diferenciales que se han pasado a través de la operación de división hacia arriba en la etapa S9640. Si el resultado del cálculo resulta ser menor que nQBitDND que se ha almacenado en el ciclo anterior de la operación de DND en la etapa S9652, reemplaza el número mínimo de bits nQBitDND requerido para la codificación después de la operación de DND en la etapa S9658.

El procesador de DND 952 realiza la operación de DND un número predeterminado de veces, y el número de ejecuciones de la operación de DND puede variar. Por ejemplo, en la presente realización, se realiza la operación de DND 7 veces. El operador de DND 952 saca nQBitDND y los datos diferenciales correspondientes a nQBitDND al primer selector de datos diferenciales 954. El operador de DND 952 saca el orden de DND por el cual se han generado los datos diferenciales correspondientes al codificador de cabecera 400 y permite su inclusión en el flujo de bits.

El primer selector de datos diferenciales 954 recibe los datos diferenciales que han pasado a través de la operación de desplazamiento, los datos diferenciales que han pasado a través de la operación de plegado, y los datos diferenciales que han pasado a través de la operación de DND y determina qué datos diferenciales de entre los tres se codificarán en entropía.

Refiriéndonos a la FIG. 10A, el primer selector de datos diferenciales 954 selecciona los resultados de la DPCM y realiza una operación de desplazamiento sobre los mismos en la etapa S9710 si el número mínimo de nQBitDND de bits requeridos para la codificación después de la operación de DND no es menor que el tamaño de nQBitDND-DPCM de bits para la codificación después de la operación de DPCM en la etapa S9700. A continuación, el primer selector de datos diferenciales 954 saca los resultados de la operación de desplazamiento al codificador de entropía 960 y permite su codificación en entropía en la etapa S9710. En este caso, el orden de la operación de DND se establece a -1, se saca al codificador de cabecera 400, y se incluye en la cabecera de claves.

Sin embargo, si en la etapa S9720 resulta que nQBitDND es menor que nQStep-DPCM y no es menor que el tamaño de bits para la codificación después de la operación de plegado, el primer selector de datos diferenciales 954 saca los datos diferenciales que se han pasado a través de la operación de plegado al codificador de entropía 960 y permite su codificación en entropía en la etapa S9730, en cuyo caso el orden de la operación de DND se establece a 0, se saca al codificador de cabecera 400, y de ese modo se incluye en la cabecera de claves.

Si el número de bits para la codificación de datos diferenciales después de la operación de DND es el menor, el primer selector de datos diferenciales 954 saca los datos diferenciales que han pasado a través de la operación de DND al operador de desplazamiento hacia arriba 956, y a continuación el operador de desplazamiento hacia arriba 956 calcula un primer grado de dispersión de la entrada de datos diferenciales desde el primer selector de datos diferenciales 954, en la etapa S9740. A continuación el operador de desplazamiento hacia arriba 956 realiza la operación de desplazamiento hacia arriba definida por la siguiente ecuación sobre los datos diferenciales que se han pasado a través de la operación de DND, en la etapa S9800, y calcula un segundo grado de dispersión de los resultados de la operación de desplazamiento hacia arriba en la etapa S9810.

$$\textit{desplazamiento hacia arriba}\ (nQKey_j, nKeyMax) \quad \ldots(16)$$

$$= nQKey_j \qquad (si\ nQKey_j \geq 0)$$

$$= nKeyMax - nQKey_j \qquad (si\ nQKey_j < 0)$$

En este punto, j indica un índice de los datos diferenciales de datos de claves cuantizados, $nQKey_j$ indica una disposición de números enteros, y nKeyMax indica un valor máximo entre los datos diferenciales.

Cuando se introducen los datos diferenciales que han pasado a través de la operación de DND y los datos diferenciales que han pasado a través de la operación de desplazamiento hacia arriba, el segundo selector de datos diferenciales 958 compara el primer grado de dispersión con el segundo grado de dispersión en la etapa S9900. Si el segundo grado de dispersión es más pequeño que el primer grado de dispersión, el segundo selector de datos diferenciales 958 saca los datos diferenciales que se han pasado a través de la operación de desplazamiento hacia arriba al codificador de entropía 960 y permite su codificación en entropía en la etapa S9910. El segundo selector de datos diferenciales 958 saca los valores máximo y mínimo nKeyMax y nKeyMin usados en la operación de DND, y el valor máximo nKeyMax usado en la operación de desplazamiento hacia arriba al codificador de cabecera 400 y permite su inclusión en la cabecera de claves.

Sin embargo, si el primer grado de dispersión es menor que el segundo grado de dispersión, el segundo selector de datos diferenciales 958 saca los datos diferenciales que se han pasado a través de la operación de DND al codificador de entropía 960 y permite su codificación en entropía en la etapa S9920. A continuación, el segundo selector de datos diferenciales 958 saca sólo los valores máximo y mínimo nKeyMax y nKeyMin usados en la operación de DND al codificador de cabecera 400, En una realización preferida de la presente invención, puede usarse la desviación estándar como una medida de los grados de dispersión primero y segundo.

El codificador de entropía 960 realiza dos funciones diferentes sobre los datos diferenciales dependiendo de las características de los datos diferenciales. Por ejemplo, los datos diferenciales que se han pasado a través de la operación de DPCM y la operación de desplazamiento y los datos diferenciales que se sólo han pasado a través de una operación de división tienen ambos valores positivos y negativos, y de este modo se requiere que realicen un procedimiento de codificación del signo de cada uno de los datos diferenciales así como los propios datos diferenciales. Por otra parte, como los datos diferenciales que se han pasado a través de la operación de plegado sólo tienen valores positivos, sólo se realiza un procedimiento de codificación de los datos diferenciales.

En una realización preferida de la presente invención, se usa la función encodeSignedAAC para codificar los datos diferenciales y también sus signos, y se usa la función encodeUnsignedAAC para codificar sólo los datos diferenciales.

La FIG. 11 es un diagrama de un ejemplo de la función encodeSignedAAC. Refiriéndonos a la FIG. 11, cuando un valor de entrada es 74, y el número de bits para codificar el valor de entrada es de 8, su signo es 0, y es el mismo que un número binario de 1001010. Los signos y todos los planos de bits se codifican del siguiente modo:

Primera etapa: se codifica un número binario sobre cada plano de bit en orden desde su bit más significativo (MSB) al bit menos significativo (LSB);

Segunda etapa: se comprueba si el bit que se está codificando actualmente es 0;

Tercera etapa: si el bit que se está codificando actualmente no es 0, el signo del número binario se codifica a continuación; y

Cuarta etapa: se codifican los bits restantes del número binario.

La función encodeUnsignedAAC codifica los valores que no tienen un signo como un flujo de bits de codificación aritmética adaptativa que usa un contexto respecto a los valores. Esta función es casi la misma que la función encodeSignedAAC excepto que existe un contexto de signo.

Las FIG. 12A hasta 12J son gráficos que muestran los datos de claves que se han sujeto a operaciones de acuerdo con una realización preferida de la presente invención. En las FIG. 12A hasta 12J, el eje – X representa los índices de cada uno de los datos de claves, y el eje Y representa los valores de los datos de claves.

La FIG. 12A es un gráfico que muestra la entrada de datos de claves originales al codificador de la presente invención. Los datos de claves mostrados en la FIG. 12A se sacan al dispositivo de cuantización 910 y a continuación se cuantizan con nueve bits de cuantización de modo que se obtienen los datos de clave cuantizados mostrados en la FIG. 12B. Si se realiza la DPCM sobre los datos de claves cuantizados mostrados en la FIG. 12B, se obtienen los datos diferenciales mostrados en la FIG. 12C.

A continuación, los datos diferenciales de los datos de claves cuantizados se desplazan usando un valor de modo de aproximadamente 7 de modo que se obtienen los datos diferenciales mostrados en la FIG. 12D. Después de esto, si se realiza una operación de plegado sobre los datos diferenciales desplazados, pueden obtenerse los datos que tienen sólo valores positivos, como se muestra en la FIG. 12E.

Los resultados de la realización de la operación de DND sobre los datos plegados mostrados en la FIG. 12E, se muestran en las FIG. 12F hasta 12H. En particular, los resultados de la realización de una operación de división sobre los datos plegados se muestran en la FIG. 12F. Como se muestra en la FIG. 12F, el intervalo de valores de datos de claves positivos va desde 0 hasta 28, y el intervalo de valores de datos de claves negativos desde -29 hasta 0, lo que significa que el intervalo de valores de los datos de claves negativos es mayor que el de los valores de datos de claves positivos. Por consiguiente no se requiere realizar la operación de división hacia arriba sobre los

datos mostrados en la FIG. 12F, y los resultados de la operación de división hacia arriba se muestran en la FIG. 12G.

Como resultado de la operación de división hacia arriba, el intervalo de valores de datos de claves negativos se reduce considerablemente de modo que es mucho más pequeño que el intervalo de valores de datos de claves positivos. En un ciclo posterior de la operación de DND, se realiza una operación de división hacia abajo sobre los resultados de la operación de división hacia arriba. La FIG. 12H es un diagrama que muestra los resultados de realizar la operación de la división hacia abajo de los datos diferenciales mostrados en la FIG. 12G. Los resultados de la realización de una operación de desplazamiento hacia arriba sobre los datos de claves mostrados en la FIG. 12H se muestran en la FIG. 12I.

Como se muestra en las FIG. 12A hasta 12G, el intervalo de datos de claves y los datos diferenciales disminuyen gradualmente. Sin embargo, como se muestra en las FIG. 12H y 12I, el intervalo de datos diferenciales se aumenta más después de la operación de desplazamiento hacia arriba que antes, lo que muestra que los datos diferenciales que se han pasado a través de la operación de división hacia abajo, como se muestra en la FIG. 12H, son los que finalmente se codifican, como se muestra en la FIG. 12J.

La información codificada en el codificador de cabecera 400 y almacenada en la cabecera de claves se describirá en lo siguiente.

Cuando se introducen los datos de claves a codificar, el codificador de cabecera 400 codifica el número de dígitos de los datos de claves y el número de claves a codificar. A continuación el codificador de cabecera 400 recibe información sobre si existe o no una región lineal de datos de claves que se ha pasado a través la codificación lineal de claves en los datos de claves de entrada y el número de datos de claves en la región lineal de datos de claves desde el codificador lineal de claves 900 y recibe los datos de claves del comienzo y del final de la región lineal de datos de claves que han pasado a través de la transformación de números de punto flotante desde el convertidor de números de punto flotante 905.

En un caso en el que el convertidor de números de punto flotante 905 recibe los valores máximo y mínimo que pueden lograr un error de cuantización mínimo y los convierte en números de punto flotante, los valores convertidos máximo y mínimo se introducen en el codificador de cabecera 400 a partir del convertidor de números de punto flotante 905 de modo que pueden usarse de nuevo para la cuantización inversa. Además, el tamaño de los bits de cuantización también se introduce al codificador de cabecera 400 y se incluye en la cabecera de clave.

El codificador de cabecera de claves 400 recibe el orden de la DPCM y los datos de intra claves en cada uno de los ciclos de DPCM desde el procesador de DPCM 920 y recibe un valor del modo que se ha usado para una operación de desplazamiento desde el dispositivo de desplazamiento 930. Además, el codificador de cabecera 400 recibe desde el procesador de DND 950 información sobre si se ha realizado o no la operación de desplazamiento hacia arriba, el orden de DND por el cual puede minimizarse el grado de la dispersión de los datos diferenciales, y los valores máximo y mínimo en cada uno de los ciclos de DND.

Finalmente, el codificador de cabecera 400 recibe el número de bits usado para la codificación desde el codificador de entropía 960 y lo codifica como una cabecera de claves.

En adelante en ese documento, se describirá la estructura y operación de un codificador de datos de valores de claves 300 de acuerdo con una primera realización de la presente invención de forma más completa con referencia a las FIG. 13A hasta 20B.

La FIG. 13A es un diagrama de bloques de un codificador de datos de valores de claves 300 de acuerdo con una primera realización de la presente invención. Refiriéndonos a la FIG 13A, el codificador de datos de valores de claves 300 incluye un operador de DPCM rotacional 1300, que calcula los valores diferenciales rotacionales de entre los valores de la transformación rotacional de un objeto en sucesivos fotogramas clave, que aplican datos de cuaternios de valores de claves de los fotogramas clave a sus objetos respectivos, cuantiza los valores diferenciales rotacionales, y saca los datos diferenciales rotacionales, un operador de DPCM circular 1400, que realiza de forma selectiva una operación de DPCM lineal y una operación de DPCM circular sobre los datos diferenciales rotacionales cuantizados, un codificador de entropía 1450, que codifica en entropía los datos diferenciales rotacionales con la DPCM rotacional o con la DPCM circular, y un codificador de cabecera 400, que codifica la información necesaria para decodificar los datos de valores de claves codificados de un nodo interpolador de orientación.

El operador de la DPCM rotacional 1300 incluye un primer multiplicador de cuaternios 1310, que calcula un valor diferencial rotacional entre un valor de la transformación rotacional de un objeto en el fotograma clave anterior y un valor de la transformación rotacional del objeto en un fotograma clave actual por la multiplicación de cuaternios del valor de la transformación rotacional del objeto en un fotograma clave actual por el valor de la transformación rotacional del objeto en el fotograma clave anterior, un dispositivo de cuantización 1340 que genera datos diferenciales rotacionales cuantizados por la cuantización no lineal de la entrada de valores diferenciales rotacionales desde el primer multiplicador de cuaternios 1310 , un dispositivo de cuantización inversa 1350, que genera datos diferenciales rotacionales restaurados del objeto en el fotograma clave actual por la cuantización inversa de los datos diferenciales rotacionales cuantizados, un segundo multiplicador de cuaternios 1370, que

restaura el valor de la transformación rotacional del objeto en el fotograma clave actual por la multiplicación de cuaternios del valor diferencial rotacional del objeto en el fotograma clave actual por los valores de la transformación rotacional del objeto en los fotograma clave anteriores, calculados acumulando los datos diferenciales rotacionales, y un retardador 1390, que saca el valor de la transformación rotacional restaurado del objeto en el fotograma clave actual para el primer multiplicador de cuaternios 1310 cuando el valor de la transformación rotacional del objeto se introduce en el siguiente fotograma clave.

En lo sucesivo se describirá con mayor detalle, una operación de DPCM rotacional realizada en un operador de DPCM rotacional de acuerdo con la presente invención.

En un procedimiento de DPCM lineal, que se ha adoptado en el MPEG-4 PMFC convencional, se calcula un valor diferencial entre un valor de la transformación rotacional de cuaternios Q_1 ($Q_1 = (q_{1,0}, q_{1,1}, q_{1,2}, q_{1,3})$) que representa la transformación rotacional de un objeto en un fotograma clave actual (o que representa el grado al cual se transforma de forma rotacional un objeto en un fotograma clave actual) y un valor de la transformación rotacional de cuaternios Q_2 ($Q_2 = (q_{2,0}, q_{2,1}, q_{2,2}, q_{2,3})$) que representa la transformación rotacional del objeto en un siguiente fotograma clave, siguiendo la ecuación (17).

$$Q_{DPCMLineal} = (q_{1,0} - q_{2,0}, q_{1,1} - q_{2,1}, q_{1,2} - q_{2,2}, q_{1,3} - q_{2,3}) \quad \ldots(17)$$

Sin embargo, el procedimiento de la DPCM lineal sólo calcula un valor diferencial entre componentes de cuaternios, que no refleja ningún valor diferencial rotacional significativo, es decir, un valor diferencial rotacional real. Por consiguiente, el procedimiento de la DPCM lineal no puede reducir la redundancia entre los datos de valores de claves sucesivos para codificar de forma eficaz. Además, en el procedimiento de la DPCM lineal, se codifican las tres componentes de un cuaternio excepto para la componente que tiene el mayor valor. Por lo tanto, es necesario codificar adicionalmente la información de 2 bits de longitud sobre la componente, que no se ha codificado, y transmitir la información de 2 bits de longitud a un decodificador desde un codificador.

Para resolver el problema anterior con el procedimiento de la DPCM lineal convencional reduciendo la redundancia entre sucesivos datos de valores de claves, y el codificador de datos de valores de claves 300 de acuerdo con la primera realización de la primera invención proporciona una operación DPCM rotacional, en la cual se consideran los valores diferenciales rotacionales reales.

Cuando $\hat{X}$ representa un vector de referencia que representa una posición de referencia de un objeto en un fotograma clave real, ($\hat{N}_{i-1}$, θ_{i-1}) representan los datos de valores de claves cuando los datos de claves son iguales a k_{i-1} , e $\hat{}y_{i-1}$ es un vector de desplazamiento de $\hat{X}$ obtenido por transformación de forma rotacional del objeto en el fotograma clave actual, la transformación rotacional en un espacio de cuaternios puede expresarse por la siguiente ecuación.

$$Y_{i-1} = Q_{i-1} \times X_0 \times Q^*_{i-1} \quad \ldots(18)$$

En la ecuación (18), X_0, Y_{i-1}, Q_{i-1}, y Q^*_{i-1} representan cuaternios unidad de $\hat{X}$, $\hat{Y}_{i-1}$, ($\hat{N}_{i-1}$, θ_{i-1}), y ($\hat{N}_{i-1}$, θ_{i-1})*, respectivamente. Además Q^*_{i-1} representa un cuaternio conjugado compuesto de Q_{i-1}, y x representa la multiplicación de cuaternios.

De este modo, cuando los datos de claves son iguales a K_i, la transformación rotacional en un espacio de cuaternios puede expresarse por la siguiente ecuación.

$$Y_i = Q_i \times X_0 \times Q_i^* \quad \ldots(19)$$

Un valor diferencial rotacional de entre los valores de transformaciones rotacionales sucesivas de datos de valores de claves sucesivos pueden calcularse con la ecuación (20) siguiente.

$$Y_i = Q_i \times X_0 \times Q_i^* = Q_i \times Q^*_{i-1} \times Y_{i-1} \times Q_{i-1} \times Q_i^* = Q_i \times Y_{i-1} \times Q_i^* \quad \ldots(20)$$

Por consiguiente, un valor diferencial rotacional puede definirse por la siguiente ecuación.

$$\dot{Q}_i = Q_i \times Q^*_{i-1} \quad \ldots(21)$$

Para impedir que un error de cuantización se propague al siguiente valor diferencial rotacional, el codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención redefine el valor diferencial rotacional definido por la ecuación (21) usando un valor de la transformación rotacional en el siguiente fotograma clave y el valor de la transformación rotacional restaurado $\hat{Q}^*_{i-1}$ en el fotograma clave anterior, que se muestra en la ecuación (22).

$$\dot{Q}_i = Q_i \times \hat{Q}^*_{i-1} \quad \ldots(22)$$

En lo sucesivo, se describirá un procedimiento para la codificación de datos de valores de claves de acuerdo con una primera realización de la presente invención con referencia a la FIG. 13B.

La FIG. 13B es un diagrama de flujo para la codificación de datos de valores de claves de acuerdo con una primera realización de la presente invención.

El codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención recibe el orden de la DPCM, un modo de codificación de entropía, y los datos de valores de claves en la etapa S13000.

A continuación, el primer multiplicador de cuaternios 1310 recibe los datos de valores de claves Q_i representados por un cuaternio y comprueba si los datos de valores de claves de entrada Q_i son los primeros datos de valores de claves Q_0 en la etapa S13050. Si los datos de valores de claves Q_i son los primeros datos de valores de claves Q_0, eso significa que no hay ningún valor de transformación de cuaternios acumulado usado para la multiplicación de cuaternios. Por consiguiente, el multiplicador de cuaternios 1310 saca los datos de valores de claves de entrada Q_i al dispositivo de cuantización 1340. Si los datos de valores de claves de entrada Q_i no son los primeros datos de valores de claves Q_0, el primer multiplicador de cuaternios 1310 calcula un valor diferencial de cuaternios Q_i, que es un valor diferencial entre los datos de valores de claves del fotograma clave actual y los datos de valores de claves del fotograma clave anterior, en la etapa S13100 por la multiplicación de cuaternios (Q_i x $\hat{Q}^*_{i-1}$) del valor de la transformación de cuaternios $\hat{Q}^*_{i-1}$ (el conjugado complejo de $\hat{Q}_{i-1}$) en el fotograma clave anterior por los datos de valores de claves de entrada Q_i representado por un valor de la transformación de cuaternios.

El dispositivo de cuantización 1340 recibe los primeros datos de valores de claves Q_0 o el valor diferencial rotacional de cuaternios Q_i desde el primer multiplicador de cuaternios 1310 y cuantiza la entrada usando un número predeterminado de bits de cuantización en la etapa S13300.

Como todos los valores rotacionales diferenciales de cuaternios se representan cada uno por un cuaternio unidad, un modelo de cuaternio que representa un valor diferencial rotacional es siempre 1. Por lo tanto, una componente de entre las cuatro componentes de un cuaternio que no se ha codificado, puede decodificarse usando los otros tres componentes. El codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención codifica sólo tres componentes de entre las cuatro componentes de un cuaternio para reducir la cantidad de datos a codificar. Por consiguiente, el dispositivo de cuantización 1340 cuantiza sólo las tres componentes que se codificarán.

El dispositivo de cuantización 1340 de acuerdo con la presente invención, realiza la cuantización no lineal en lugar de la cuantización lineal general, y la razón se describirá más adelante con referencia a la FIG. 14A.

La FIG. 14A es un diagrama que ilustra un ejemplo típico de una función de distribución de probabilidad (PDF) en cada una de las componentes de un valor diferencial rotacional. Como se muestra en la FIG. 14A, los valores componentes de los valores diferenciales de rotación generalmente se concentran alrededor de 0, lo que se llama compactación de energía y significa que es posible reducir la redundancia entre elementos de la información de rotación de forma eficaz. Por consiguiente, para realizar la cuantización de los valores diferenciales rotacionales, que reflejan los valores de componentes más bajos de cada uno de los valores diferenciales rotacionales de forma más suficiente, se necesita realizar elaboradamente la cuantización sobre los valores de las componentes más bajas, y esto es por lo que el dispositivo de cuantización 1340 de acuerdo con la presente invención realiza una cuantización no lineal. En este punto, el dispositivo de cuantización 1340 usa una curva arco tangente para asignar un factor de escala no lineal a cada uno de los valores diferenciales rotacionales.

La FIG. 14B es una curva de arco-tangente usada para la cuantización de acuerdo con la presente invención. Como se muestra en la FIG. 14B, la curva de arco tangente proporciona una resolución apropiada para valores de entrada más altos y proporciona una resolución mucho más alta para valores de entrada más bajos. El dispositivo de cuantización 1340 cuantiza un valor diferencial rotacional en la etapa S13330 que usa una función de escala no lineal, lo que se muestra en la siguiente ecuación.

$$\begin{aligned} \tilde{q}_i &= floor(q_i' \cdot (2^{nQBits-1} - 1) + 0.5) \quad (q_i' \geq 0) \quad \cdots(23) \\ &= -floor(-q_i' \cdot (2^{nQBits-1} - 1) + 0.5) \quad (q_i' < 0) \\ (q_i' &= \frac{4}{\pi} \cdot \tan^{-1}(q_i), i = 1, 2, 3, -1 \leq q_i \leq 1) \end{aligned}$$

En este punto, q_i representa cada una de los componentes de la entrada de los valores diferenciales rotacionales al dispositivo de cuantización 1340, q_i' representa un valor escalado de q_i, nQBits representa un número predeterminado de bits de cuantización usados para la cuantización, y floor (x) representa una función para transformar el valor de entrada x en un número entero máximo no mayor que x.

El dispositivo de cuantización 1340 realiza la cuantización sobre los valores diferenciales rotacionales de entrada y a continuación comprueba si los datos diferenciales rotacionales cuantizados corresponden o no a los últimos datos de valores de claves a codificar. Si los datos diferenciales rotacionales cuantizados corresponden a los últimos datos de valores de claves, el dispositivo de cuantización 1340 saca los datos diferenciales rotacionales cuantizados al operador de DPCM circular 1400. Si los datos diferenciales rotacionales cuantizados no corresponden a los últimos datos de valores de claves, el dispositivo de cuantización 1340 saca los datos diferenciales rotacionales cuantizados al dispositivo de cuantización inversa 1350.

En la etapa S13400, el dispositivo de cuantización inversa 1350 realiza la cuantización inversa de la entrada de datos diferenciales rotacionales cuantizados desde el dispositivo de cuantización 1340 y saca los valores diferenciales rotacionales restaurados en la etapa S13500.

Como se ha descrito anteriormente, el codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención codifica sólo tres componentes de un cuaternio que representan un valor diferencial rotacional por la cuantización de las tres componentes distintas de la primera componente. Un dispositivo de cuantización inversa 2430 de un decodificador de datos de valores de claves, que se muestra en la FIG. 21A, y el dispositivo de cuantización inversa 1350, que se ha descrito anteriormente, se supone que restauran la otra componente que no se ha codificado, de entre las cuatro componentes del valor diferencial rotacional, usando las tres componentes codificadas. Como todos los valores diferenciales rotacionales se representan cada uno por un cuaternio unidad, el modelo de un cuaternio que representa un valor diferencial rotacional es siempre 1. Por lo tanto, es posible restaurar la primera componente de entre las cuatro componentes del valor diferencial rotacional usando la siguiente ecuación.

$$\hat{q}_0 = \sqrt{1-(\hat{q}_1^2+\hat{q}_2^2+\hat{q}_3^2)} \quad (\hat{q}_0 \geq 0) \qquad \ldots(24)$$

En la ecuación (24) $\hat{}q_1$, $\hat{}q_2$, y $\hat{}q_3$ representan las tres componentes restauradas de un valor diferencial rotacional, y $\hat{q}_0$ representa la primera componente restaurada usando las tres componentes restauradas $\hat{q}1$, $\hat{q}2$, y $\hat{q}3$.

Para restaurar la primera componente $\hat{q}_0$ siguiendo la siguiente ecuación (11), la primera componente $\hat{q}_0$, debe tener un valor positivo, y esta condición se puede satisfacer aprovechando las características de un cuaternio, que aparecen cuando se aplica un cuaternio a la transformación rotacional de un objeto en un espacio de 3D y se muestra en la ecuación (25).

$$\boldsymbol{Y = Q \times X \times Q^* = (-Q) \times X \times (-Q)^*} \qquad \ldots(25)$$

La ecuación (25) muestra que los valores de transformación rotacional Q y –Q tienen el mismo significado físico cuando se aplica la transformación rotacional a un objeto en un espacio en 3D. Por consiguiente, si la primera componente de un valor diferencial rotacional en el codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención tiene un valor negativo, el valor diferencial rotacional puede convertirse fácilmente en un número positivo multiplicando cada una de las componentes del valor diferencial rotacional por -1. En este caso, sin embargo, la suma de los cuadrados de cada una de las componentes de un valor diferencial rotacional restaurado excepto para la primera componente puede exceder de 1 debido a un error de cuantización. En este caso $\hat{q}_0$ no puede determinarse por la ecuación (24) y se considera un valor que está cerca de 0 y es más pequeño que un valor mínimo que se puede cuantizar por el dispositivo de cuantización 1340. El hecho de que $\hat{q}_0$ tenga tal valor significa que el objeto se ha transformado de forma rotacional tanto como aproximadamente 180 grados. Por consiguiente, un aparato para la decodificación de un interpolador de orientación necesita un procedimiento para la determinación de $\hat{q}_0$ mientras que minimiza el efecto del primer valor de la componente restaurada $\hat{q}_0$ sobre los otros tres componentes restaurados $\hat{q}_1$, $\hat{q}_2$, y $\hat{q}_3$, y el procedimiento se aplicará también al dispositivo de cuantización inversa 1350 del codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención. Por ejemplo, el valor mínimo, que se puede cuantizar por el dispositivo de cuantización 1340, multiplicado por un número entero predeterminado puede determinarse como $\hat{}q_0$. Puede usarse la siguiente ecuación para determinar $\hat{q}_0$.

$$\hat{q}_0 = a \cdot 2^{-m} \qquad \ldots(26)$$

En la ecuación (26) 'a' representa una constante arbitraria, y m representa un número predeterminado de bits de cuantización.

Mientras se requiere una información de 2 bits de longitud sobre cada uno de los datos de valores de claves para la

decodificación en el MPEG-4 BIFS convencional, el procedimiento PMFC no necesita la información de 2 bits de longitud en la presente invención, y de este modo el número de bits a codificar se puede disminuir tanto como en 2N cuando se codifican N datos de valores de claves.

La salida de valores diferenciales rotacionales restaurados desde el dispositivo de cuantización inversa 350 se introduce al segundo multiplicador de cuaternios 1370, y el segundo multiplicador de cuaternios 1370 comprueba si el fotograma clave de entrada es el primer fotograma clave a codificar en la etapa S13600. Si el fotograma clave de entrada es el primer fotograma clave a codificar, el segundo multiplicador de cuaternios 1370 acumula el valor diferencial rotacional restaurado de la entrada del primer fotograma clave desde el dispositivo de cuantización inversa 1350 $(\hat{Q}_0 = \hat{\tilde{Q}}_0)$ y saca el valor acumulado al retardador 1390 en la etapa S13800.

Si el fotograma clave de entrada no es el primer fotograma clave a codificar, el segundo multiplicador de cuaternios1370 restaura un valor de la transformación rotacional $\hat{Q}_i$ de un fotograma clave actual en la etapa S13700 por una multiplicación de cuaternios de un valor diferencial rotacional $\hat{\tilde{Q}}_i$ del fotograma clave actual y un valor de la transformación rotacional restaurado $\hat{Q}_{-1}$ de un fotograma clave anterior $(\hat{Q}_i = \hat{\tilde{Q}}_i \times \hat{Q}_{i-1})$.

El segundo multiplicador de cuaternios 1370 saca el valor de la transformación rotacional restaurado $\hat{Q}_i$ del fotograma clave actual al retardador 1390, y a continuación el retardador 1390 mantiene $\hat{Q}_i$ hasta que se introduce un valor de la transformación rotacional de un fotograma clave siguiente de modo que $\hat{Q}_i$ es igual a $\hat{Q}_{i-1}$ y a continuación saca $\hat{Q}_{i-1}$ al primer multiplicador de cuaternios 1310.

Cuando un valor de la transformación rotacional Q_i de un fotograma clave actual se introduce al primer multiplicador de cuaternios 1310, el primer multiplicador de cuaternios también recibe el valor de la transformación rotacional restaurado $\hat{}Q_{i-1}$ de un fotograma clave anterior desde el retardador 1390 y genera un valor diferencial rotacional Q_i entre el valor de la transformación rotacional del fotograma clave actual y el valor restaurado de la transformación rotacional del fotograma clave anterior en la etapa S13100 por la multiplicación de cuaternios del valor restaurado de la transformación del cuaternio ^Q*i-1 en el fotograma clave anterior Qi. (Qi x ^Q*i-1).

El valor diferencial rotacional generado se saca al dispositivo de cuantización 340 y a continuación se cuantiza por el dispositivo de cuantización 1340 en la etapa S13300, como se ha descrito anteriormente. El dispositivo de cuantización 1340 saca los datos diferenciales rotacionales generados por la cuantización de un valor diferencial rotacional al operador de DPCM circular 1400 en la etapa S13400 si los valores diferenciales rotacionales de entrada corresponden a los últimos datos de valores de claves.

El operador de la DPCM circular 1400 comprueba si el orden de DPCM de la entrada de datos diferenciales rotacionales cuantizados desde el dispositivo de cuantización 1340 es 0. Si el orden de la DPCM es 0, el operador de la DPCM circular 1400 saca los datos diferenciales rotacionales cuantizados al codificador de entropía 1450 en la etapa S14000 sin realizar una operación de DPCM lineal y una operación de DPCM circular. Si el orden de la DPCM no es 0, el operador de la DPCM circular 1400 realiza una operación de DPCM lineal y una operación de DPCM circular sobre los datos diferenciales rotacionales cuantizados en la etapa S14100.

La FIG. 15A es un diagrama que ilustra un ejemplo de salida de datos diferenciales desde el dispositivo de cuantización 1340, y la FIG. 15B es un diagrama que ilustra los resultados de realizar una operación de DPCM lineal sobre la salida de datos diferenciales desde el dispositivo de cuantización 1340.

Como se muestra en la FIG. 15B, como resultado de una operación de DPCM lineal, el intervalo de datos diferenciales a codificar puede aumentarse el doble de lo que se suele aumentar. El propósito de la actuación de la operación de DPCM circular es mantener el intervalo de los datos diferenciales dentro del intervalo de los datos diferenciales cuantizados.

La operación de DPCM circular se realiza sobre la suposición de que un valor máximo y un valor mínimo en un intervalo de cuantización están conectados circularmente entre sí. Por consiguiente, si los datos diferenciales, que son el resultado de la realización de la DPCM lineal sobre dos datos cuantizados consecutivos, son mayores que la mitad del valor máximo en el intervalo de cuantización, pueden representarse por valores más pequeños restando el valor máximo de los datos diferenciales.

Si los datos diferenciales son menores que la mitad del valor mínimo en el intervalo de cuantización, pueden representarse por valores incluso más pequeños añadiendo el valor máximo en el intervalo de cuantización a los datos diferenciales.

Cuando $\tilde{Q}_1$ y $\tilde{Q}_{i-1}$ representan datos diferenciales rotacionales cuantizados en dos momentos sucesivos de tiempo t_i y t_{i-1}, respectivamente, se realiza una operación de DPCM lineal sobre los dos datos diferenciales rotacionales cuantizados sucesivos $\tilde{Q}_i$ y $\tilde{Q}_{i-1}$ siguiendo la ecuación (27).

$$X_i = \tilde{Q}_i - \tilde{Q}_{i-1} = (\tilde{q}_{i,1} - \tilde{q}_{i-1,1},\ \tilde{q}_{i,2} - \tilde{q}_{i-1,2},\ \tilde{q}_{i,3} - \tilde{q}_{i-1,3})^{\gamma} \quad \ldots(27)$$

Además, se realiza una operación de DPCM circular sobre los datos diferenciales usando la ecuación (27), siguiendo la ecuación (28).

$$\tilde{Q}_i = \min(|X_i|, |X'_i|) \quad \ldots(28)$$

$$X'_i = X_i - (2^{nQBits} - 1) \quad \textbf{(si } Xi \geq \textbf{0)}$$

$$X'_i = X_i + (2^{nQBits} - 1) \quad \textbf{(en caso contrario)}$$

En la ecuación (28), nQBits representa un número predeterminado de bits de cuantización. La FIG. 15C es un diagrama que ilustra los resultados de la realización de una operación de DPCM circular sobre los datos diferenciales con la DPCM mostrados en la FIG. 15B. Como se muestra en la FIG. 15C, el intervalo de los datos diferenciales con la DPCM circular es mucho más pequeño que el intervalo de los datos diferenciales con la DPCM lineal.

El codificador de entropía 1450 recibe los datos diferenciales rotacionales o los datos diferenciales rotacionales con la DPCM circular desde el operador de DPCM circular 1400 dependiendo del orden de la DPCM sobre los datos diferenciales rotacionales y a continuación codifica los datos diferenciales de entrada eliminando los bits de redundancia.

Refiriéndonos de nuevo a la FIG. 13B, el decodificador de entropía 1450 comprueba el modo de codificación de entropía en la etapa S14500.

Si el modo de codificación de entropía es un modo de codificación de entropía binario, el codificador de entropía 1450 codifica los datos diferenciales de entrada usando una función SignedAAC () en la etapa S14600. Si el modo de codificación de entropía es un modo de codificación de entropía unario, el codificador de entropía 1450 codifica los datos diferenciales de entrada usando una función UnaryAAC () en la etapa S14700.

La función SignedAAC () se usa para codificar los datos diferenciales usando un codificador aritmético binario adaptativo que codifica el signo y la magnitud de los datos diferenciales sobre cada uno de los planos de bits, y el procedimiento de codificación se ha descrito anteriormente con referencia a la FIG. 11.

Por otra parte, la función UnaryAAC ()se usa para codificar símbolos transformando un valor a codificar en una serie de ceros, un bit de bandera '1' que indica el final de la serie de ceros, y otro bit que indica el signo del valor. El número de la serie de ceros corresponde a la magnitud del valor.

En lo sucesivo, se describirá la función UnryAAC () más completamente con referencia a la FIG. 16B, Por ejemplo, 256 se codifica en una serie de bits consistente de doscientos cincuenta y seis ceros, un 1 que es un bit de bandera que indica el final de la serie de ceros, y un 0 representa el signo de 256, es decir, un signo más por la función UnaryAAC ().

Debido a la función UnaryAAC (), aumenta la redundancia entre los bits que representan el símbolo a codificar, lo que mejora la eficacia de la codificación de símbolos.

En lo sucesivo, se describirá un codificador de datos de valores de claves 300 de acuerdo con una segunda realización de la presente invención. El codificador de datos de valores de claves 300 de acuerdo con la segunda realización de la presente invención incluye un medio para corregir un error de la dirección de rotación que ocurre durante la cuantización así como todos los elementos del codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención.

La FIG. 17 es un diagrama que ilustra un error de la dirección de rotación generado durante la codificación de los valores de la transformación rotacional de cuaternios usando valores diferenciales rotacionales. El error de la dirección de rotación se produce debido a que el procedimiento de codificación de cuaternios realizado en el codificador de datos de valores de claves 300 de acuerdo con la presente invención es un procedimiento de codificación de pérdidas.

En la FIG. 17 cuando Q_i, Q_{i-1}, $\hat{Q}_i$ y $\hat{Q}_{i-1}$ representan un valor de la transformación rotacional que se aplica actualmente a un objeto en un fotograma clave de orden i, el valor de la transformación rotacional aplicada al objeto en un fotograma clave anterior, el valor restaurado desde Q_i y el valor restaurado desde Q_{i-1} respectivamente, las localizaciones del objeto girado así como las transformaciones de rotación indicadas respectivamente por Q_i, Q_{i-1}, $\hat{Q}_i$, y $\hat{Q}_{i-1}$ pueden representarse por cuatro áreas diferentes, como se muestra en la FIG. 17.

En otras palabras, si el objeto gira desde Q_{i-1} a Q_i a lo largo del arco más corto de modo que el objeto está localizado en el área 1 y el área 3 después de girar por Q_i y Q_{i-1}, se considera que el objeto ha girado desde Q_{i-1} a Q_i en el sentido contrario a las agujas de reloj. Entre tanto, si el objeto gira desde Q_{i-1} a Q_i a lo largo de la trayectoria más corta de modo que el objeto está localizado en el área 2 y el área 4 después del giro por Q_{i-1} y Q_i, se considera que el objeto ha girado desde Q_{i-1} a Q_i en el sentido de las agujas de reloj.

Por el contrario, si el objeto gira siguiendo la información de rotación que se ha codificado y se ha decodificado a continuación, un aparato para la decodificación de un interpolador de orientación gira el objeto usando $\hat{Q}$ y $\hat{Q}_{i-1}$ correspondientes a Q_i y Q_{i-1}, respectivamente. Por consiguiente, si la localización relativa del objeto girado por $\hat{Q}_i$ con respecto a la localización del objeto girado por $\hat{Q}_{i-1}$ es el área 2 ó 3 $\hat{Q}_{i-1}$, el objeto gira en el sentido contrario a las agujas de reloj. Si la localización relativa del objeto girado por ^Q_i con respecto a la localización del objeto girado por $\hat{Q}_{i-1}$ es el área 1 ó 4, el objeto gira en el sentido de las agujas de reloj. En las áreas 1 y 2, la dirección de rotación del objeto cuando usa un valor de la transformación rotacional original puede ser opuesto a la dirección de rotación del objeto cuando usa un valor de la transformación rotacional decodificado, debido a que Q_i y $\hat{Q}_i$ son diferentes debido a la codificación con pérdidas, que se realiza para codificar los valores de la transformación rotacional de cuaternios. Para resolver este problema, se requiere minimizar el grado al cual gira el objeto en una dirección equivocada, que es opuesta a la dirección deseada, o para corregir la dirección de rotación del objeto cuando el objeto gira en una dirección equivocada de modo que el objeto gire en una dirección deseada. En la presente invención, se adopta el procedimiento para corregir la dirección de rotación del objeto de modo que se haga la rotación del objeto en la dirección deseada.

Refiriéndonos de nuevo a la FIG. 17, se describirá brevemente el concepto de la corrección de un error de la dirección de rotación de acuerdo con la presente invención en lo siguiente. Si se detecta un error de la dirección de rotación, como el fenómeno que ocurre en las áreas 1 y 2, los valores diferenciales rotacionales de cuaternios a codificar se controlan de modo que giran en una dirección correcta, en cuyo caso la inconsistencia en las direcciones de rotación aún se produce en el área 2. Sin embargo en el área 2, a diferencia del área 1, las diferencias entre los valores de cuaternios originales y los valores de cuaternios restaurados son relativamente pequeñas. Por consiguiente, la corrección de la dirección de rotación de acuerdo con la segunda realización de la presente invención se realiza sólo en el área 1.

En lo sucesivo, se describirá el codificador de datos de valores de claves 300, que realiza la corrección de la dirección de rotación, de acuerdo con la segunda realización de la presente invención con referencia a las FIG. 18A hasta la 19B. El codificador de datos de valores de claves 300 de acuerdo con la segunda realización de la presente invención tiene casi la misma estructura que el codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención. La única diferencia entre ellos es la estructura de un operador de DPCM rotacional, y de ese modo sólo se describirá la estructura de un operador DPCM rotacional en el codificador de datos de valores de claves 300 de acuerdo con la segunda realización de la presente invención en los siguientes párrafos.

La FIG. 18A es un diagrama de bloques de un operador de DPCM rotacional 1800 de acuerdo con la segunda realización de la presente invención, y la FIG. 18B es un diagrama de bloques de un calculador de errores de la dirección de rotación 1820 mostrados en la FIG. 18A.

Refiriéndonos a la FIG. 18A, el operador de DPCM rotacional 1800 incluye un calculador de errores de la dirección de rotación 1820, que recibe un valor de la transformación rotacional de un objeto en un fotograma clave actual y un valor restaurado de la transformación rotacional del objeto en un fotograma clave anterior y calcula los errores de la dirección de rotación, un detector de errores de la dirección de rotación 1830, que detecta en base a la entrada de errores de la dirección de rotación desde el calculador de errores de la dirección de rotación 1820 si ha ocurrido o no un error suficiente para cambiar la dirección de rotación del objeto durante la decodificación, un corrector de la dirección de rotación 1815, que corrige y saca un valor diferencial rotacional introducido desde el primer multiplicador de cuaternios 1810 de modo que el objeto, que está localizado adaptando una transformación de un valor de la transformación rotacional decodificado del fotograma clave anterior, puede girar además tanto como 180 grados en la dirección de rotación original, y un selector de la dirección de rotación 1835, que selecciona la entrada de valores diferenciales rotacionales desde el corrector de la dirección de rotación 1815 o la entrada del valor diferencial rotacional desde el primer multiplicador de cuaternios 1810 dependiendo del valor introducido desde el detector de errores de la dirección de rotación 1830 y saca el valor seleccionado al dispositivo de cuantización 840.

Refiriéndonos a la FIG. 18B, el calculador de errores de la dirección de rotación 1820 mostrado en la FIG. 18A incluye un retardador 1822, que almacena un valor de la transformación rotacional de entrada hasta que se introduce el valor de la transformación rotacional del fotograma clave siguiente, un tercer multiplicador de cuaternios 1824, que recibe el valor de la transformación rotacional de entrada y un valor de la transformación rotacional de un salida del fotograma clave anterior desde el retardador 1822 y calcula un valor diferencial rotacional entre el valor de la transformación rotacional de entrada y el valor de la transformación rotacional del fotograma clave anterior por la

multiplicación de cuaternios, un cuarto multiplicador de cuaternios 1826, que calcula un valor diferencial rotacional entre el valor de la transformación rotacional de la salida del fotograma clave anterior desde el retardador 1822 y el valor restaurado de transformación rotacional del fotograma clave anterior, y un quinto multiplicador de cuaternios 1828 que calcula entre el valor de la transformación rotacional de entrada y el valor restaurado de la transformación rotacional del fotograma clave anterior.

En lo sucesivo, se describirá con mayor detalle una operación de DPCM rotacional de acuerdo con una segunda realización de la presente invención con referencia a la FIG. 19A.

Los datos de valores de claves Q_i de un nodo interpolador de orientación, que están próximos a codificarse, se introducen en el primer multiplicador de cuaternios 1810 y el calculador de errores de la dirección de rotación 1820 del operador de DPCM rotacional 1800 en la etapa S19000.

El primer multiplicador de cuaternios 1810, como el primer multiplicador de cuaternios en el codificador de datos de valores de claves 300 de acuerdo con la primera realización de la presente invención, genera un valor diferencial rotacional Q_i por la multiplicación de cuaternios ($Q_i \times \hat{Q}^*_{i-1}$) un valor de la transformación rotacional de entrada de un fotograma clave actual por un valor de la transformación rotacional restaurado de una entrada del fotograma clave anterior desde el segundo multiplicador de cuaternios 1870 y saca el valor diferencial rotacional generado Q_i al corrector de la dirección de rotación 1815 y el selector de la dirección de rotación 1835 en la etapa S19100.

El corrector de la dirección de rotación 1815 corrige la entrada al mismo del valor de la dirección rotacional siguiendo la ecuación (29) y saca el valor diferencial rotacional corregido Q_s al selector de la dirección de rotación 1835. El calculador de errores de la dirección de rotación 1820 recibe el valor de la transformación rotacional Q_i del fotograma clave actual y el valor restaurado de la transformación rotacional $\hat{Q}_{i-1}$ de la entrada del fotograma clave anterior desde el segundo multiplicador de cuaternios 1870 y calcula los valores de la transformación rotacional Q_A, Q_B, y Q_C, que se describirán más adelante. El detector de errores de la dirección de rotación 1830 detecta usando la entrada de valores diferenciales rotacionales desde el calculador de errores de la dirección de rotación 1820 si ha ocurrido o no un error de la dirección de rotación y saca el resultado de la detección al selector de la dirección de rotación 1835 en la etapa S19200.

$$Q_s = \begin{pmatrix} \dfrac{|\delta_T|}{-\dfrac{1}{\sqrt{(\dot{q}_{R,1})^2 + (\dot{q}_{R,2})^2 + (\dot{q}_{R,3})^2}}\dot{q}_{R,1}} \\ -\dfrac{1}{\sqrt{(\dot{q}_{R,1})^2 + (\dot{q}_{R,2})^2 + (\dot{q}_{R,3})^2}}\dot{q}_{R,2} \\ -\dfrac{1}{\sqrt{(\dot{q}_{R,1})^2 + (\dot{q}_{R,2})^2 + (\dot{q}_{R,3})^2}}\dot{q}_{R,3} \end{pmatrix} \quad \ldots(29)$$

En la ecuación (29), δ_T representa una constante muy próxima a 0 y $(q_{R,0}, q_{R,1}, q_{R,2}, q_{R,3})^T$ representa la salida del valor diferencial rotacional Q_i desde el primer multiplicador de cuaternios 1810. El calculador de errores de la dirección de rotación 1820, el detector de errores de la dirección de rotación 1830, y el corrector de la dirección de rotación 1815 se describirán más adelante con referencia a la FIG. 19B.

El selector de la dirección de rotación 1835 comprueba si ha ocurrido o no un error de la dirección de rotación de modo que el objeto gira en la dirección opuesta a la dirección deseada cuando se decodifican los datos de valores de claves codificados usando una entrada de valores lógicos desde el detector de errores de la dirección de rotación 1830. Si no se ha producido un error de la dirección de dirección, el selector de la dirección de rotación 1835 saca la entrada del valor diferencial de rotación desde el primer multiplicador de cuaternios 1810 al dispositivo de cuantización 1840 en la etapa S19300. Si ha ocurrido un error de la dirección de rotación, el selector de la dirección de rotación 1835 saca la entrada del valor diferencial rotacional correcto desde el corrector de la dirección de rotación 1815 en la etapa S19300.

El dispositivo de cuantización 1840 cuantiza los datos diferenciales rotacionales originales Q_i o los datos diferenciales rotacionales corregidos Q_s en las etapas S19400 y S19500 en el mismo procedimiento de cuantización que en la primera realización de la presente invención.

El dispositivo de cuantización 1840 comprueba si los datos diferenciales rotacionales cuantizados $\tilde{Q}_i$ pertenecen a los últimos datos de valores de claves en la etapa S19600. Si los datos diferenciales rotacionales cuantizados $\tilde{Q}_i$ corresponden a los últimos datos de valores de claves, el dispositivo de cuantización 1840 saca los datos diferenciales rotacionales cuantizados $\tilde{Q}_i$ a un operador de DPCM circular 1400 en la etapa S19700. Si los datos diferenciales rotacionales cuantizados $\tilde{Q}_i$ no corresponden a los últimos datos de valores de claves, el dispositivo de cuantización 1840 saca los datos diferenciales rotacionales cuantizados $\tilde{Q}_i$ al dispositivo de cuantización inversa 1850.

El dispositivo de cuantización inversa 1850 realiza la cuantización inversa de los datos diferenciales rotacionales cuantizados $\tilde{Q}_i$ en el mismo procedimiento de cuantización inversa que en la primera realización de la presente invención y saca los datos diferenciales rotacionales restaurados $\hat{\hat{Q}}_i$ al segundo multiplicador de cuaternios 1870 en la etapa S19800.

El segundo multiplicador de cuaternios 1870 genera un valor restaurado de la transformación rotacional $\hat{Q}_i$ de un fotograma clave actual por la multiplicación de cuaternios del valor diferencial rotacional restaurado $\hat{\hat{Q}}_i$ por un valor de la transformación rotacional $\hat{Q}_{i-1}$ de un fotograma clave anterior. El segundo multiplicador de cuaternios 870 saca un valor de la transformación rotacional de un fotograma clave actual al primer multiplicador de cuaternios 1810 y el calculador de errores de la dirección de rotación 1820 a través del retardador 1890 en la etapa S19900.

En lo sucesivo, se describirán las operaciones del calculador de errores de la dirección de rotación 1820, el detector de errores de la dirección de rotación 1830, y el corrector de la dirección de rotación 1815 con referencia la FIG. 19B.

El calculador de errores de la dirección de rotación 1820 calcula los valores diferenciales rotacionales correspondientes a las áreas de 2 hasta 4, que se han descrito anteriormente con relación a la FIG. 17.

El calculador de errores de la dirección de rotación 1820 recibe un valor de la transformación de rotación, que corresponde a los datos de valores de claves de un fotograma clave a codificar, recibe un valor restaurado de la transformación rotacional de un fotograma clave anterior al fotograma clave actual, y calcula un error de la dirección de rotación en la etapa S19220.

Los valores de la transformación rotacional correspondientes a los datos de valores claves de un interpolador de orientación del fotograma clave actual se introducen directamente en el tercer multiplicador de cuaternios 1824 y el quinto multiplicador de cuaternios 1828 y se introducen en el cuarto multiplicador de cuaternios 1826 a través del retardador 1822 cuando se introduce un valor de la transformación rotacional del siguiente fotograma clave. Además, los valores restaurados de la transformación rotacional de la salida del fotograma clave anterior desde el segundo multiplicador de cuaternios 1870 se introducen en el cuarto multiplicador de cuaternios 1826 y el quinto multiplicador de cuaternios 1828.

El tercer multiplicador de cuaternios 1824 genera el valor diferencial rotacional Q_A por la multiplicación de cuaternios ($Q_A = Q_i \times Q^*_{i-1}$) el valor de la transformación rotacional del fotograma clave actual por el valor de la transformación rotacional del fotograma clave anterior y saca Q_A al detector de errores de la dirección de rotación 1830. El valor diferencial rotacional Q_A representa una dirección rotacional original de un objeto durante un intervalo de tiempo [t_{i-1}, t_i].

El cuarto multiplicador de cuaternios 1826 genera el valor diferencial rotacional Q_B por la multiplicación de cuaternios ($Q_B = Q_{i-1} \times \hat{Q}^*_{i-1}$) el valor de la transformación rotacional del fotograma clave anterior por el valor restaurado de la transformación rotacional del fotograma clave anterior y saca Q_B al detector de errores de la dirección de rotación 1830. El valor diferencial rotacional Q_B calculado por el cuarto multiplicador de cuaternios 1826 representa el error de la dirección de rotación y la dirección de rotación de un objeto, que están afectadas por un error de cuantización, en un momento de tiempo predeterminado t_{i-1} y corresponde al área 2 mostrada en la FIG. 17.

El quinto multiplicador de cuaternios 1828 genera el valor diferencial rotacional Q_C por la multiplicación de cuaternios ($Q_C = Q_i \times \hat{Q}^*_{t-1}$) el valor de la transformación rotacional del fotograma clave actual por el valor de la transformación rotacional restaurado del fotograma clave anterior y saca Q_C al detector de errores de la dirección de rotación 1830. El valor diferencial rotacional Q_C calculado por el quinto multiplicador de cuaternios 1828 representa un valor diferencial rotacional a codificar en el momento predeterminado de tiempo t_i y corresponde al área 4 mostrada en la FIG. 17.

El detector de errores de la dirección de rotación 1830 comprueba usando la entrada de los valores diferenciales rotacionales Q_A, Q_B, y Q_C desde el calculador de errores de dirección de rotación 1820 si el objeto gira o no en la dirección opuesta a la dirección original en el procedimiento de decodificación de los datos de valores de claves en

la etapa S19240. Para hacer esto, el detector de errores de la dirección de rotación 1830 comprueba si los valores diferenciales rotacionales Q_A, Q_B y Q_C satisfacen las desigualdades (30 hasta (34). En primer lugar, el detector de errores de la dirección de rotación 1830 comprueba si los valores diferenciales rotacionales Q_A y Q_C satisfacen la desigualdad (30).

$$\begin{pmatrix} q_{A,1} \\ q_{A,2} \\ q_{A,3} \end{pmatrix} \cdot \begin{pmatrix} q_{C,1} \\ q_{C,2} \\ q_{C,3} \end{pmatrix} < 0 \qquad \ldots(30)$$

En la desigualdad (30), cuando la entrada del valor diferencial rotacional Q_A desde el calculador de errores de la dirección de rotación 1820 se expresa por $Q_A = (q_{A,0}, q_{A,1}, q_{A,2}, q_{A,3})^T$, $\begin{pmatrix} q_{A,1} \\ q_{A,2} \\ q_{A,3} \end{pmatrix}$ representa un vector tridimensional $(q_{A,1}, q_{A,2}, q_{A,3})^T$ consistente de las cuatro componentes $q_{A,0}$, $q_{A,1}$, $q_{A,2}$, y $q_{A,3}$ de Q_A excepto para la primera componente $q_{A,0}$. $\begin{pmatrix} q_{C,1} \\ q_{C,2} \\ q_{C,3} \end{pmatrix}$ representa un vector tridimensional $(q_{C,1}, q_{C,2}, q_{C,3})^T$ consistente de cuatro componentes $q_{C,0}$, $q_{C,1}$, $q_{C,2}$ y $q_{C,3}$ de Q_C excepto para la primera componente $q_{C,0}$. La desigualdad (30) muestra una condición de que un producto interno de dos vectores de 3D $(q_{A,1}, q_{A,2}, q_{A,3})$ y $(q_{C,1}, q_{C,2}, q_{C,3})^T$ es menor que cero.

Si el producto interno en la desigualdad (30) es menor que cero, la dirección de rotación de un objeto cuando se usa Q_A es opuesta a la dirección de rotación del objeto cuando se usa Q_C, Cuando el producto interior es menor que cero, el valor de la expresion lógica mostrada en la desigualdad (30) se fija a 'cierto'. Cuando el producto interno es mayor que cero, el valor de la expresión lógica se fija a 'falso'

$$\begin{pmatrix} q_{B,1} \\ q_{B,2} \\ q_{B,3} \end{pmatrix} \cdot \begin{pmatrix} q_{C,1} \\ q_{C,2} \\ q_{C,3} \end{pmatrix} < 0 \qquad \ldots(31)$$

En la desigualdad (31), cuando el valor diferencial rotacional Q_B entra desde el calculador de errores de la dirección de rotación 1820 se expresa por $Q_B = (q_{B,0}, q_{B,1}, q_{B,2}, q_{B,3})^T$, $\begin{pmatrix} q_{B,1} \\ q_{B,2} \\ q_{B,3} \end{pmatrix}$ representa un vector tridimensional $(q_{B,1}, q_{B,2}, q_{B,3})^T$ consistente de cuatro componentes $q_{B,0}$, $q_{B,1}$, $q_{B,2}$, y $q_{B,3}$ de Q_B excepto para la primera componente $q_{B,0}$. $\begin{pmatrix} q_{C,1} \\ q_{C,2} \\ q_{C,3} \end{pmatrix}$ representa un vector tridimensional $(q_{C,1}, q_{C,2}, q_{C,3})^T$ consistente de los cuatro componentes $q_{C,0}$, $q_{C,1}$, $q_{C,2}$ y $q_{C,3}$ de Q_C excepto para la primera componente $q_{C,0}$. La desigualdad (31) muestra una condición de que un producto interno de dos vectores de 3D $(q_{B,1}, q_{B,2}, q_{B,3})^T$ y $(q_{C,1}, q_{C,2}, q_{C,3})^T$ es menor que cero.

Si el producto interno en la desigualdad (31) es menor que cero, la dirección de rotación de un objeto cuando se usa Q_B es opuesta a la dirección de rotación del objeto cuando se usa Q_C, Cuando el producto interior es menor que cero, el valor de la expresion lógica mostrada en la desigualdad (31) se fija a 'cierto'. Cuando el producto interno es mayor que cero, el valor de la expresión lógica se fija a 'falso'

$$A_{TH} < 2\cos^{-1}\left|q_{A,0}\right| \qquad \ldots(32)$$

En la desigualdad (32), $q_{A,0}$ representa la primera componente del valor diferencial rotacional Q_A, y A_{TH} se fija a una constante predeterminada próxima a cero. Cuando en la desigualdad (32), $2.\cos^{-1} |q_{A,0}|$ es mayor que A_{TH}, el valor de la expresión lógica mostrada en la desigualdad (32) se define como 'cierto'. Cuando $2.\cos^{-1} |q_{A,0}|$ no es mayor A_{TH}, el valor de la expresión lógica mostrada en la desigualdad (32) se define como 'falso'. Cuando la desigualdad

(32) se define como 'falso', significa que el ángulo de rotación θ = ($2.\cos^{-1}|q_{A,0}|$) menor que la constante predeterminada A_{TH} puede ignorarse cuando se realiza un codificador de datos de valores de claves 300 de acuerdo con la presente invención. Incluso aunque existe un error de la dirección de rotación tan grande como el ángulo de rotación θ, el error de la dirección de rotación no causa imágenes con una distorsión severa para los ojos humanos, especialmente cuando se usa un procedimiento de medición del error propuesto por la presente invención.

$$A_{TH} < 2\cos^{-1}|q_{B,0}| \quad \ldots(33)$$

En la desigualdad (33), $q_{B,0}$ representa la primera componente del valor diferencial rotacional Q_B, y A_{TH} es la misma que la correspondiente a la desigualdad (32). Cuando en la desigualdad (33), $2.\cos^{-1}|q_{B,0}|$ es mayor que A_{TH}, el valor de la expresión lógica mostrada en la desigualdad (33) se define como 'cierto'. Cuando $2.\cos^{-1}|q_{B,0}|$ no es mayor A_{TH}, el valor de la expresión lógica se define como 'falso'.

$$A_{TH} < 2\cos^{-1}|q_{C,0}| \quad \ldots(34)$$

En la desigualdad (34), $q_{C,0}$ representa la primera componente del valor diferencial rotacional Q_C, y A_{TH} es la misma que la correspondiente a la desigualdad (32). Cuando en la desigualdad (34), $2.\cos^{-1}|q_{C,0}|$ es mayor que A_{TH}, el valor de la expresión lógica mostrada en la desigualdad (34) se define como 'cierto'. Cuando $2.\cos^{-1}|q_{C,0}|$ no es mayor A_{TH}, el valor de la expresión lógica se define como 'falso'.

El detector de errores de la dirección de rotación 1830 realiza una operación de AND sobre los valores lógicos de las desigualdades (30) hasta (34) y saca el resultado de la operación AND al selector de la dirección de rotación 1835.

El primer multiplicador de cuaternios 1810 tiene dos valores de entrada Q_i y $\hat{Q}_{i-1}$ en un momento predeterminado de tiempo t. El primer multiplicador de cuaternios 1810 saca un valor diferencial rotacional usando los dos valores de entrada Q_i y $\hat{Q}_{i-1}$. Como se ha descrito anteriormente, en el área 1, el aparato para la decodificación de los datos de valores de claves codificados, que recibe los datos diferenciales rotacionales, gira el objeto en el sentido de las agujas de reloj. Sin embargo, el objeto debe girar desde el lugar donde está localizado actualmente después de girar en el número de grados predeterminado indicado por Q_{i-1} a un lugar donde se supone que está localizado después de la rotación en el número de gados predeterminado indicado por Q_i, y de ese modo la dirección de rotación original del objeto debe ser en la dirección contraria a las agujas de reloj.

Por consiguiente, el corrector de la dirección de rotación 1815 corrige la dirección de rotación del objeto de modo que el objeto puede girar tanto como el ángulo de rotación indicado por $\hat{Q}_{i-1}$en la misma dirección que ha girado en base a $\hat{Q}_i$, como se muestra en la FIG. 17, es decir, de modo que el objeto puede girar en el sentido contrario a las agujas de reloj desde el lugar donde está localizado actualmente después de girar en base $\hat{Q}_{i-1}$ al sitio donde se supone que estará localizado después de girar tanto como el ángulo de rotación indicado por $\hat{Q}_{i.}$

Para hacer esto, el corrector de la dirección de rotación 1815 establece la nueva información de rotación para hacer que el objeto gire en el sentido contrario a las agujas de reloj tanto como 180 grados desde el sitio donde está situado actualmente después de girar el número predeterminado de grados que sigue a $\hat{Q}$i-1. Por consiguiente, la dirección de rotación del objeto se corrige para que sea la misma que la original y el error de la dirección de rotación puede minimizarse. El corrector de dirección de rotación 1815 recibe los datos diferenciales rotacionales Qi del fotograma clave actual desde el primer multiplicador de cuaternios 1810, genera los datos diferenciales rotacionales que tienen una dirección de rotación corregida, y saca los datos diferenciales rotacionales corregidos al selector de la dirección de rotación 1835 en la etapa S19260.

Refiriéndonos a la FIG 19A, el selector de la dirección de rotación 1835 comprueba si la entrada de valores lógicos desde el detector de errores de la dirección de rotación 1830 son ciertos en la etapa S19300. Si los valores lógicos de la entrada son ciertos, el selector de la dirección de rotación 1835 determina que ha ocurrido el mismo fenómeno que ocurrió en el área 1 mostrado en la FIG. 17 y saca el valor diferencial rotacional corregido Q_S definido por la ecuación (29) al dispositivo de cuantización 1840 en la etapa S19400.

Por el contrario, si los valores lógicos de entrada son falsos el selector de la dirección de rotación 1835 determina que el mismo fenómeno que ocurrió en el área 1 no ha ocurrido y saca la entrada del valor diferencial rotacional Q_i desde el primer multiplicador de cuaternios 1810 al dispositivo de cuantización 1840 en la etapa S19500.

En lo siguiente, se describirá un codificador de datos de valores de claves 300 de acuerdo con una tercera realización de la presente invención.

Como los codificadores de datos de valores de claves 300 de acuerdo con la primera y segunda realizaciones de la presente invención codifican sólo tres componentes de entre los cuatro componentes de un valor diferencial rotacional, no son capaces de restaurar el valor de la primera componente del cuaternio usando la ecuación (24) debido al error de cuantización. Por ejemplo, una primera componente de un valor diferencial rotacional restaurado puede ser un número imaginario.

Para impedir el problema con la codificación de sólo tres componentes de un cuaternio, el codificador de datos de valores de claves 300 de acuerdo con la tercera realización de la presente invención incluye un dispositivo de cuantización, que puede ajustar de forma apropiada tres valores componentes cuantizados de modo que cuando se decodifican los datos de valores de claves codificados, la otra componente se puede restaurar a un número positivo real y la distorsión de las imágenes puede minimizarse.

El codificador de datos de valores de claves 300 de acuerdo con la tercera realización de la presente invención es el mismo que los codificadores de datos de valores de claves 300 de acuerdo con la primera o la segunda realizaciones de la presente invención excepto para la estructura de un dispositivo de cuantización 2000, y de ese modo sólo se describirá la estructura del dispositivo de cuantización 2000 en lo siguiente.

La FIG. 20A es un diagrama de bloques de un dispositivo de cuantización 2000 incluido en el codificador de datos de valores de claves 300 de acuerdo con la tercera realización de la presente invención. Refiriéndonos a la FIG. 20A, el dispositivo de cuantización 2000 incluye una unidad de cuantización 2010, que cuantiza un valor diferencial rotacional introducido en el mismo, un ajustador de datos de cuantización 2020, que ajusta un valor diferencial rotacional cuantizado, un dispositivo de cuantización inversa 2030, que realiza la cuantificación inversa de los datos diferenciales rotacionales cuantizados, un restaurador de valores diferenciales rotacionales 2040, que restaura todos los valores de componentes de los datos diferenciales rotacionales cuantizados restaurando una primera componente de los datos diferenciales rotacionales cuantizados usando los componentes cuantizados en inverso, y una unidad de medición de errores 2050, que mide un error entre un valor diferencial restaurado y un valor diferencial rotacional introducido originalmente y renueva los datos diferenciales rotacionales cuantizados.

La FIG. 20B es un diagrama de flujo de la operación del dispositivo de cuantización 2010. Refiriéndonos a la FIG. 20B, cuando se introduce un valor diferencial rotacional $\dot{Q}$ desde un primer multiplicador de cuaternios, el dispositivo de cuantización 2010 cuantiza el valor diferencial rotacional de entrada $\dot{Q}$ usando la ecuación (23) y saca los datos diferenciales rotacionales cuantizados $\tilde{Q}$ al ajustador de datos de cuantización 2020 en la etapa S20050.

El ajustador de datos de cuantización 2020 ajusta las tres componentes de los datos diferenciales rotacionales cuantizados $\tilde{Q}$ siguiendo la ecuación (35) en la etapa S20100.

$$I_{ijk} = \tilde{Q} + D_{ijk}\ ,\ D_{ijk} = (i, j, k)^r \qquad (-d \le i, j, k \le d \text{ donde } i, j, k \text{ y } d \text{ son números enteros}) \qquad \ldots(35)$$

En la ecuación (35), i, j, y k son variables, que se añadirán a los datos diferenciales rotacionales de modo que ajustan los datos diferenciales rotacionales, y se usan para definir el intervalo de i, j y k. Las tres componentes ajustadas de los datos diferenciales rotacionales cuantizados $\tilde{Q}$ se sacan al dispositivo de cuantización inversa 2030.

El dispositivo de cuantización inversa 2030 realiza la cuantización inversa de los datos diferenciales cuantizados rotacionales ajustados I_{ijk} y saca el valor diferencial rotacional con la cuantización inversa $\hat{\tilde{I}}_{ijk}$ (o los resultados de la cuantización inversa) al restaurador de valores diferenciales rotacionales 2040 en la etapa S2020.

El restaurador de valores diferenciales rotacionales 2040, que recibe tres componentes del valor diferencial rotacional cuantizado inverso $\hat{\tilde{I}}_{ijk}$, restaura una primera componente de los datos diferenciales rotacionales siguiendo la ecuación (24) y saca un valor diferencial rotacional restaurado a la unidad de medición de errores 2050 en la etapa S20300.

La unidad de mediciones de error 2050 comprueba un valor de la primera componente de un valor diferencial rotacional introducido en la misma. Si el valor de la primera componente es un número real, la unidad de mediciones de error 2050 realiza la medición de error. Por el contrario, si el valor de la primera componente es un número imaginario, el procedimiento se mueve a la etapa S20600 en la etapa S20400.

Cuando la primera componente del valor diferencial rotacional de entrada es un número real, la unidad de mediciones de error 2050 mide un error e_{ijk} entre un valor diferencial rotacional original y un valor diferencial rotacional restaurado en la etapa S20450 y comprueba si e_{ijk} es menor que un error ajustado $e_{i^*j^*k^*}$ en la etapa S20500. El procedimiento de medición de e_{ijk} entre el valor diferencial rotacional original y el valor diferencial rotacional restaurado se describirá más adelante con referencia a la FIG. 27.

Si el error medido e_{ijk} es menor que el error ajustado e_{i*j*k*}, e_{ijk} reemplaza a e_{i*j*k*} en la etapa S20550 y a continuación se comprueba si las variables i, j, y k pertenecen a un intervalo de ajuste [-d, +d], en la etapa S20600. Si las variables i, j, k pertenecen a intervalo de ajuste [-d, +d] la unidad de mediciones de error 2040 realiza repetidamente las etapas S20100 hasta S20550. En la etapa S20100 durante cada uno de los ciclos de las etapas S20100 hasta S20550, se añade 1 a las componentes de la segunda hasta la cuarta de los datos diferenciales rotacionales cuantizados en una forma de bucle anidado.

Por ejemplo, un ajustador de datos cuantizados 2020 mantiene las componentes segunda y tercera, intenta hacer que la primera componente restaurada se convierta en un valor real añadiendo un valor de k al valor de la cuarta componente mientras que aumenta gradualmente el valor de k añadiendo continuamente 1 al valor de k ($-d \leq k \leq d$) antes de que el valor de k exceda +d, y a continuación encuentra los valores ajustados de las cuatro componentes, que pueden minimizar un error entre un valor diferencial rotacional de entrada y un valor diferencial rotacional restaurado.

Si el valor de k alcanza +d, el ajustador de datos cuantizados 1020, intenta hacer que la primera componente restaurada se convierta en un valor real inicializando los valores de k con –d y añadiendo un valor de j al valor de la tercera componente mientras que aumenta el valor de j añadiendo 1 al valor de j ($-d \leq j \leq d$) y añadiendo un valor de k al valor de la cuarta componente mientras que aumenta gradualmente el valor de k añadiendo continuamente 1 al valor de k ($-d \leq k \leq d$) antes de que el valor de k exceda +d, y a continuación busca valores ajustados de las cuatro componentes, que pueden minimizar el error entre el valor diferencial rotacional de entrada y un valor diferencial rotacional restaurado.

Si el valor de j alcanza +d, el ajustador de datos cuantizados 1020, intenta hacer que la primera componente restaurada se convierta en un valor real inicializando los valores de j y k con –d y añadiendo un valor de i al valor de la segunda componente mientras que aumenta el valor de i añadiendo 1 al valor de i ($-d \leq i \leq d$) y añadiendo un valor de j al valor de la tercera componente y añadiendo un valor de k al valor de la cuarta componente mientras que aumenta gradualmente el valor de k añadiendo continuamente 1 al valor de k ($-d \leq k \leq d$) antes de que el valor de k exceda +d, y a continuación busca valores ajustados de las cuatro componentes, que pueden minimizar el error entre un valor diferencial rotacional de entrada y un valor diferencial rotacional restaurado.

Se repite esto hasta que i, j y k alcanzan +d y a continuación encuentra valores ajustados de las cuatro componentes, que pueden minimizar el error entre un valor diferencial rotacional de entrada y un valor diferencial rotacional restaurado.

La unidad de mediciones de error 2050 comprueba si el error ajustado e_{i*j*k*} es menor que el error final $\dot{e}_{qi*qj*qk*}$ en la etapa S20700 mientras que se cambian los valores de las componentes segunda a cuarta. Si e_{i*j*k*} es menor que $\dot{e}_{gi*gj*gk*}$, $\acute{e}_{j*j*k*}$ reemplaza a $\dot{e}_{gi*gj*gk*}$ y los datos diferenciales rotacionales cuantizados se corrigen siguiendo la ecuación (36) en la etapa S20750.

$$\tilde{Q}^* = (\tilde{q}_1, \tilde{q}_2, \tilde{q}_3)^{\gamma} + (i^*, j^*, k^*)^{\gamma} \quad \ldots(36)$$

Después de esto, la unidad de mediciones de error 2050 saca los datos diferenciales rotacionales corregidos al ajustador de datos cuantizados 2020.

El ajustador de datos cuantizados 2020 fija los valores de las variables i, j y k a –d y realiza las etapas S20100 hasta S20600 de nuevo sobre los datos diferenciales rotacionales de entrada en el mismo. A continuación, el ajustador de datos cuantizados 2020 comprueba si existen datos diferenciales rotacionales que tienen un error menor con los datos rotacionales de entrada que el error final almacenado anteriormente.

Si el error ajustado e_{i*j*k*} no es menor que el error final $e_{qi*qj*qk*}$, la unidad de mediciones de error 2050 saca los datos diferenciales rotacionales cuantizados $\tilde{Q}^{q^*} = \tilde{Q}^* = (\tilde{q}_1^*, \tilde{q}_2^*, \tilde{q}_3^*)^T$ correspondientes al error final almacenado actualmente $e_{gi*gj*gk*}$, a un operador de DPCM circular en la etapa S20800.

En lo sucesivo, se describirán más completamente un aparato y un procedimiento para la decodificación de un flujo de bits, dentro del cual se codifica el interpolador de orientación, de acuerdo con una realización preferida de la presente invención con referencia a las FIG. 21A hasta 25.

La FIG. 21A es un diagrama de bloques de un aparato para la decodificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención, y la FIG. 21B es un diagrama de flujo de un procedimiento para la codificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención.

Refiriéndonos a la FIG. 21A, el aparato para la decodificación de un interpolador de orientación incluye un decodificador de datos de claves 2120, que decodifica los datos de claves desde un flujo de bits de entrada, un decodificador de datos de valores de claves 2150, que decodifica los datos de valores de claves a partir del flujo de bits de entrada, un decodificador de cabecera 2110, que decodifica la información de cabecera a partir del flujo de

bits de entrada y saca la información de cabecera decodificada al decodificador de datos de claves 2120 y al decodificador de datos de valores de claves 2150, y un sintetizador del interpolador de orientación 2180, que sintetiza los datos de claves decodificados y los datos de valores de claves decodificados y de este modo saca un interpolador de orientación decodificado.

Refiriéndonos a la FIG. 21B, el flujo de bits codificado por el aparato para la codificación de un interpolador de orientación mostrado en la FIG. 3 se introduce al decodificador de datos de claves 2120, el decodificador de datos de claves 2150, y el decodificador de cabecera 2110 en la etapa S21000.

El decodificador de cabecera 2110 decodifica la información de cabecera a partir del flujo de bits de entrada y proporciona la información de cabecera decodificada al decodificador de datos de claves 2120, el decodificador de datos de valores de claves 2150, y el sintetizador del interpolador de orientación 2180 en la etapa S21100.

El decodificador de datos de claves 2120 decodifica en entropía los datos de claves a partir del flujo de bits de entrada, genera los datos de claves decodificados realizando una operación de DND inversa predeterminada, una operación de plegado inverso, y una operación de desplazamiento inverso, y saca los datos de claves decodificados al sintetizador del interpolador de orientación 2180. El decodificador de datos de valores de claves 2150 decodifica en entropía los datos de valores de claves a partir del flujo de bits de entrada, genera un valor diferencial rotacional usado para girar un objeto tanto como la diferencia entre las transformaciones rotacionales aplicadas el objeto por los datos de valores de claves de cuaternios de cada uno de los fotograma clave por los datos diferenciales rotacionales decodificados de cuantización inversa, restaura un valor de la transformación rotacional del fotograma clave actual por la multiplicación de cuaternios de los valores diferenciales rotacionales del fotograma clave actual por el valor de la transformación rotacional de un fotograma clave anterior, y saca el valor de la transformación rotacional del fotograma clave actual al sintetizador del interpolador de orientación 2180 en la etapa S21200.

El sintetizador del interpolador de orientación 2180 restaura un interpolador de orientación sintetizando los datos de claves y los datos de valores de claves introducidos al mismo siguiendo un modo de generación de puntos de ruptura y un indicador de selección de claves, que se introducen desde el decodificador de cabecera 2110, y saca el interpolador de orientación restaurado en la etapa S21400.

En lo sucesivo, se describirá el decodificador de datos de claves 2120 de acuerdo con una realización preferida de la presente invención y un procedimiento de decodificación de los datos de claves de acuerdo con una realización preferida de la presente invención con referencia a las FIG. 22 hasta la 23B.

La FIG. 22 es un diagrama de bloques del decodificador de datos de claves 2120. El decodificador de datos de claves 2120 recibe un flujo de bits codificado y lo reconstituye dentro de los datos de claves por decodificación.

El decodificador de datos de claves 2120 incluye un decodificador de entropía 2260, un procesador de datos DND inverso 2250, un procesador de plegado inverso 2240, un dispositivo de desplazamiento inverso 2230, un procesador de DPCM inverso 2220, un dispositivo de cuantización inversa 2210, un decodificador de claves lineal 2200, y un convertidor inverso de números de punto flotante 2205.

La FIG. 23A es un diagrama de flujo de un procedimiento para la decodificación de los datos de claves de acuerdo con una realización preferida de la presente invención. Refiriéndonos a las FIG. 22 y 23A, un flujo de bits, dentro del cual se comprimen los datos de claves, se introduce en el decodificador de cabecera 2110, y el decodificador de entropía 2260.

El decodificador de cabecera 2110 decodifica los elementos de información requeridos para cada una de las etapas de decodificación y les proporciona a sus etapas correspondientes de decodificación en la etapa S23000. La información decodificada por el decodificador de cabecera 2110 se describirá con cada una de las etapas de decodificación.

El decodificador de entropía 2260 recibe el número de datos diferenciales a decodificar y el número de bits que se han usado para la codificación, es decir, el número de bits a usar para la decodificación, desde el decodificador de cabecera 2110 y decodifica el flujo de bits de entrada en la etapa S23100. El número de datos diferenciales es igual al resultado de restar el número de datos de intra claves obtenidos realizando la DPCM del número de datos de claves.

El decodificador de entropía 2260 identifica si los datos diferenciales a decodificar tienen valores negativos o valores positivos en base a la información predeterminada incluida en el flujo de bits, por ejemplo bSignedAACFlag en la presente realización. Si los datos diferenciales codificados tienen valores negativos, el decodificador de entropía 2260 los decodifica usando la función decodeSignedAAC (). Por el contrario, si los datos diferenciales codificados tienen sólo valores positivos, el decodificador de entropía 2260 los decodifica usando la función decodeUnsignedAAC (). Después de esto, los datos diferenciales decodificados se transmiten al procesador de DND inversa 2250.

El procesador de DND inversa 2250 recibe el orden de DND y un valor máximo de nKeyMax en cada uno de los ciclos de DND desde el decodificador de cabecera 2110.

Si el orden de DND es -1, esto significa que los datos diferenciales codificados que se están decodificando se han decodificado en entropía pasando a través de una operación de DPCM y una operación de desplazamiento en lugar de pasar a través de la DND, y el procedimiento va directamente sobre la etapa de realización de una operación de desplazamiento inverso. Si el orden de la DND es 0, esto significa que los datos diferenciales codificados que se están decodificando se han decodificado en entropía pasando a través de una operación de plegado en lugar de pasar a través de la DND, y de ese modo el procedimiento va directamente a la etapa de realización de una operación de plegado inverso. Si el orden de DND es mayor que 0, se realiza una operación de DND inversa en la etapa S23200.

El procesador de la DND inversa 2250 determina si los datos diferenciales codificados que se están decodificando se han codificado pasando o no a través de una operación de desplazamiento hacia arriba en la etapa S23300. En una realización preferida de la presente invención, se determina si los datos diferenciales codificados que se están decodificando se han codificado pasando o no a través de una operación de desplazamiento hacia arriba comprobando si KeyInvertDown incluida en el flujo de bits es o no mayor que cero.

Si los datos diferenciales codificados que se están decodificando no han pasado a través de la operación de desplazamiento hacia arriba, el procedimiento va a la etapa de realización de una DND inversa. Por el contrario, si los datos diferenciales codificados que se están decodificando han pasado a través de una operación de desplazamiento hacia arriba, los datos diferenciales que se han transferido desde una región de números positivos a una región de números negativos realizando una operación de desplazamiento hacia arriba se mueven de nuevo a la región de números negativos en la etapa S23400. En una realización preferida de la presente invención, los datos diferenciales que han pasado a través de una operación de desplazamiento hacia arriba se restauran realizando una operación de desplazamiento hacia abajo (una operación de inversión hacia abajo) (invert – down) que se expresa por la siguiente ecuación.

$$\textbf{\textit{inversión hacia abajo}}\ (\textbf{\textit{v}}) \qquad \ldots(37)$$

$$= \textbf{\textit{v}} \qquad (\textbf{\textit{si v} ≤ \textit{nKey inversión hacia abajo}})$$

$$= \textbf{\textit{nkeyInvertDown – v}} \qquad (\textbf{\textit{si v} > \textit{nKey inversión hacia abajo}})$$

En este punto, nKeyInvertDown tiene el mismo valor que el valor máximo nKeyMax usado en la operación de desplazamiento hacia arriba. Como resultado de la operación del desplazamiento hacia abajo, los datos diferenciales que tiene un valor por encima de nKeyInvertDown se convierten a valores negativos por debajo de -1.

Una operación inversa de división hacia abajo o una operación inversa de división hacia arriba se realiza de forma selectiva sobre los datos diferenciales que se han pasado a través de la operación de desplazamiento hacia abajo dependiendo del valor máximo nKeyMax en cada uno de los ciclos de DND.

Refiriéndonos a la FIG. 23B, el procesador de la DND inversa 2250 realiza una operación de DND inversa tantas veces como los datos diferenciales hayan pasado a través de la operación de DND durante la codificación. En otras palabras, el procesador de la DND inversa 2250 establece un valor inicial del orden de la DND inversa que es igual al orden de la DND. A continuación, el procesador de la DND inversa 2250 resta 1 del valor inicial del orden de la DND inversa cada vez que realiza una operación de DND inversa y permanece realizando la operación de DND inversa hasta que el orden de la DND inversa se hace 1. El procesador de la DND inversa 2250 busca la nKeyMax en cada uno de los ciclos de DND y comprueba si cada uno de nKeyMax es o no menor que 0 en la etapa S23510.

Si nKeyMax es menor que 0, significa que se ha realizado una operación de división hacia arriba en el procedimiento de codificación, y de este modo el procesador de la DND inversa 2250 extiende el intervalo de datos diferenciales que se están decodificando a una región de números negativos realizando una operación inversa de división hacia arriba, en la etapa S23530. En una realización preferida de la presente invención, puede usarse una operación inversa de división hacia arriba (inverse – divide – up (v) que se define por la ecuación (38).

$$\textbf{\textit{inversión de división hacia arriba (v)}} \qquad \ldots(38)$$

$$= \textbf{\textit{v}} \qquad (\textbf{si v} \geq \textbf{0})$$

$$= (\textbf{\textit{nKeyMax}}_{\textbf{\textit{i}}} \textbf{ - 1}) - (\textbf{\textit{v}} \textbf{ - 1}) / \textbf{2} \qquad (\textbf{\textit{si v < 0, v mod}}\ \textbf{2} \neq \textbf{0})$$

$$= \textbf{\textit{v}} / \textbf{2} \qquad (\textbf{\textit{si v <0, v mod}}\ \textbf{2} = \textbf{0})$$

Sin embargo, si nKeyMax no es menor que 0, el procesador de la DND inversa 2250 comprueba si el orden de la DND inversa es 1. Si el orden de la DND no es 1, significa que se ha realizado una operación de división hacia abajo sobre los datos diferenciales que se están decodificando en el procedimiento de codificación, y de este modo, el procesador de la DND inversa 2250 extiende el intervalo de los datos diferenciales a una región de números positivos realizando una operación de división inversa hacia abajo, en la etapa S23570.

En una realización preferida de la presente invención, puede usarse una operación de división hacia abajo inversa (inverse - divide - down) que se define por la siguiente ecuación.

$$\textbf{inversión de división hacia abajo } (v) \quad \ldots(39)$$

$$= v \qquad (\text{si } v \geq 0)$$

$$= (nKeyMax_i + 1) + (v - 1) / 2 \qquad (\text{si } v < 0,\ v \bmod 2 \neq 0)$$

$$= v / 2 \qquad (\text{si } v < 0,\ v \bmod 2 = 0)$$

Si nKeyMax no es menor que 0 y el orden de la DND inversa es 1, el procesador de la DND inversa 2250 completa una operación de DND inversa entera después de la realización de una operación de división inversa en la etapa S23590. En una realización preferida de la presente invención, se puede usar una operación de división inversa (inverse divide) que se define por la ecuación (40).

$$\textbf{inversión de división } (v) \quad \ldots(40)$$

$$= v \qquad (\text{si } v \geq 0)$$

$$= v + (nKeyMax_0 + 1) \qquad (\text{si } v < 0)$$

Los datos diferenciales de los datos de claves que han pasado a través de la operación inversa de DND se introducen en el procesador de plegado inverso 2240, y el procesador de plegado inverso 2240 realiza una operación de plegado inverso sobre los datos diferenciales de modo que los datos diferenciales que suelen estar sólo en una región de números positivos se dividen en valores positivos y valores negativos en la etapa S23600. En una realización preferida de la presente invención, puede usarse una operación de plegado inverso (inverse fold) que se define por la ecuación (41).

$$\textbf{plegado inverso } (v) \quad \ldots(41)$$

$$= -(v + 1) / 2 \qquad (\text{si } v \bmod 2 \neq 0)$$

$$= v / 2 \qquad (\text{si } v \bmod 2 = 0)$$

$$= 0 \qquad (\text{si } v = 0)$$

Los datos diferenciales que se han pasado a través de la operación de plegado inverso se sacan al dispositivo de desplazamiento inverso 2230, y el dispositivo de desplazamiento inverso 2230 añade un modo nKeyShift, que se ha usado en el procedimiento de codificación y se introduce desde el decodificador de cabecera 2110, a la entrada de datos diferenciales desde el procesador de plegado inverso 2240, en la etapa S23700. Esta operación se expresa por la siguiente ecuación.

$$\textbf{plegado – shift } (v) = v + nKeyShift \quad \ldots(42)$$

El procesador de DPCM inversa 2220 restaura la entrada de datos diferenciales desde el dispositivo de desplazamiento inverso 2230 en datos de claves cuantizados usando el orden de la DPCM introducido desde el codificador de cabecera 2110, en la etapa S23800. El dispositivo de desplazamiento inverso 2230 realiza una operación de DPCM inversa tantas veces como el orden de la DPCM siguiendo la ecuación (43).

$$v(i + 1) = v(i) + delta(i) \quad \ldots(43)$$

En este punto, i indica un índice de los datos diferenciales y los datos de claves, v indica una disposición de números enteros, y delta (i) indica datos diferenciales.

Los datos de claves cuantizados que se han pasado a través de la operación de DPCM inversa se introducen en el dispositivo de cuantización inversa 2210. A continuación, el dispositivo de cuantización inversa 2210 recibe información sobre si se codifican o no el tamaño de nKeyBits de bits de cuantización y los valores máximo y mínimo usados para la cuantización inversa por el convertidor de números en punto flotante 905 desde el decodificador de cabecera 2110 y convierte los datos de claves cuantizados en datos de claves con cuantización inversa en la etapa S23900 usando la siguiente ecuación.

$$inverse - quantize(v) = fKeyMin + \frac{v}{2^{nKeyQBit} - 1} \times (fKeyMax - fKeyMin) \quad \cdots(44)$$

Si los valores máximo y mínimo usados para la cuantización no se han convertido por el conversor de números en punto flotante 905 en el procedimiento de codificación de datos, fKeyMin y fKeyMax mostrados en la ecuación (44) se fijan a 0 y a 1, respectivamente. Sin embargo, si los valores máximo y mínimo usados para la cuantización se han

convertido por el conversor de números de punto flotante 905, los valores máximo y mínimo que se convierten inversamente por el convertidor inverso de números de punto flotante 2205 se usan como los valores máximo y mínimo, respectivamente, para la cuantización inversa.

Un ejemplo de códigos de programa dentro del cual se realiza la operación de DND inversa para la cuantización inversa se describirá más adelante.

La salida de datos de claves decodificados desde el dispositivo de cuantización inversa 2210 se añaden a los datos de claves decodificados en el decodificador de claves lineal 2200, constituyendo de este modo los datos de claves decodificados.

En lo sucesivo, se describirá un procedimiento de decodificación de claves lineal a continuación.

El decodificador de cabecera 2110 decodifica la información de la cabecera de claves desde un flujo de bits. Si existe información sobre una región de datos de claves lineal en el flujo de bits, el decodificador de cabecera 2110 saca la información requerida para la decodificación de las claves de comienzo y de final de la región de datos de claves lineal al convertidor inverso de números de punto flotante 2205 y saca el número de claves, que se codifican como claves lineales, al decodificador de claves lineal 2200.

El convertidor inverso de números de punto flotante 2205 convierte inversamente las claves de comienzo y de final de la región de datos de claves lineal, que se expresan por números decimales, en números binarios y saca los números binarios al decodificador de claves lineal 2200.

Suponiendo que los dos números de punto flotante a decodificar se denominen como fKeyMin y fKeyMax, el procedimiento de decodificación de fKeyMin es como sigue.

El decodificador de cabecera 2110 lee el número de dígitos de fkeyMin desde el flujo de bits. Si el número de dígitos de fKeyMin es 0, fKeyMin se fija a 0, y el número de dígitos de fkeyMax se lee desde el flujo de bits para decodificar fKeyMax. Si el número de dígitos de fKeymax no es menor de 8, significa que fKeyMax se ha codificado siguiendo la Normativa de IEEE 754. De este modo, el número en punto flotante fKeyMax se decodifica después de que se leen 32 bits del mismo.

Sin embargo, si el número de dígitos de fKeyMax está entre 1 y 7, el decodificador de cabecera 2110 lee un bit de signo desde el flujo de bits. En una realización preferida de la presente invención, si el bit de signo es 1, MinKeyMantissaSign se fija a -1. Por el contrario, si el bit de signo es 0, MinKeyMantissaSign se fija a 1. Después de esto, el número de bits requerido para la decodificación se obtiene refiriéndonos a la Tabla 1 que muestra la relación entre el número de dígitos de una mantisa y el número de bits requerido para la codificación. A continuación, se leen tantos bits del flujo de bits como el número de bits requerido para la codificación y se almacenan en nMinKeyMantissa. A continuación, se lee el siguiente bit del flujo de bits y se almacena en MinKeyExponentSig del mismo modo que el signo de la mantisa es almacena en MinkeyMantissaSign. Los siguientes seis bits del flujo de bits, que corresponden a un valor de exponente, se leen y se almacenan en nMinKeyExponent.

El convertidor inverso de números en punto flotante 2205 restaura fkeyMin sustituyendo el valor de entrada desde el

decodificador de cabecera 2110 en la ecuación (45).

$$fKeyMin = \frac{MinKeyMantissaSign * nMinKeyMantissa}{10^{MinKeyExponentSign * nMinKeyExponent}} \quad \ldots(45)$$

El procedimiento de restauración de fkeyMax es el mismo que el procedimiento de restauración de fKeyMin. En particular, se determina si se usa o no el mismo valor que el exponente de fKeyMin como exponente de fKeyMax antes de leer el exponente de fKeyMax del flujo de bits. Si no se usa el mismo valor que el exponente de fKeyMin como el exponente de fKeyMax, se lee el exponente de fKeyMax del flujo de bits del mismo modo que se lee el exponente de fKeyMin del flujo de bits.

El decodificador de claves lineal 2200 recibe las claves de comienzo y de final de la región de datos de claves lineal desde el convertidor inverso de números de punto flotante 2205 y decodifica la región de datos de claves lineal siguiendo la ecuación (46).

$$Key_i = fKeyMin + \frac{(fKeyMax - fKeyMin) * i}{(nNumberOfLinearKey - 1)}$$
$$(i = 0, \ldots, nNumberOfLinearKey - 1) \quad \ldots(46)$$

En este punto, fKeyMin y fKeyMax indican los datos de claves de comienzo y de final, respectivamente, de la región de datos de claves lineal.

Los datos de claves en la región de datos de claves lineal decodificados usando el procedimiento mencionado anteriormente se añaden a la salida de datos de claves desde el dispositivo de cuantización inversa 2210, y a continuación se sacan los resultados de la adición como datos de claves finales.

En lo sucesivo, se describirá un aparato para la decodificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención y un procedimiento para la codificación de los datos de valores de claves de acuerdo con una realización preferida de la presente invención con referencia a las FIG. 24A hasta 25.

La FIG. 24A es un diagrama de bloques de un aparato para la decodificación de un interpolador de orientación de acuerdo con una realización preferida de la presente invención.

Refiriéndonos a la FIG. 24A, el aparato para la decodificación de un interpolador de orientación incluye un decodificador de cabecera 2110, que decodifica la información de cabecera requerida para la decodificación de los datos de valores de claves representados por un cuaternio desde un flujo de bits de entrada y proporciona la información de la cabecera decodificada al decodificador de datos de valores de claves 2150, un decodificador de entropía 2410 que genera datos diferenciales rotacionales con la DPCM circular o datos diferenciales rotacionales cuantizados por la decodificación de entropía de los datos de valores de claves codificados en entropía desde el flujo de bits de entrada, un operador de DPCM circular inversa 2420, que genera datos diferenciales rotacionales cuantizados realizando una operación de DPCM circular inversa sobre los datos diferenciales rotacionales con la DPCM circular introducidos en el mismo, un dispositivo de cuantización inversa 2430, que genera un valor diferencial rotacional por la cuantización inversa de los datos diferenciales rotacionales cuantizados, y un multiplicador de cuaternios 2440, que genera un valor de la transformación rotacional, de un fotograma clave actual por la multiplicación de cuaternios de un valor diferencial rotacional del fotograma clave actual por el valor de la transformación rotacional de un fotograma clave anterior.

La FIG. 24B es un diagrama de flujo de un procedimiento para la decodificación de datos de valores de claves de acuerdo con una realización preferida de la presente invención. Refiriéndonos a la FIG. 24B, se introduce un flujo de bits, dentro del cual se codifican los datos de valores de claves usando el aparato para la codificación de un interpolador de orientación de acuerdo con la presente invención, dentro del decodificador de cabecera 2110 y el decodificador de entropía 2410 del decodificador de datos de valores de claves 2150.

El decodificador de cabecera 2110 decodifica la información de cabecera requerida para la decodificación de los datos de valores de claves desde el flujo de bits de la entrada y proporciona la información de cabecera decodificada al decodificador de datos de valores de claves 2150 en la etapa S24100.

En la información de cabecera, se codifican los primeros y segundos datos de valores de intra claves cuantizados en cuaternios en base al orden de la DPCM, y una pluralidad de indicadores usados para decodificar otros datos de valores de claves.

Si el orden de la DPCM es 1 (por ejemplo, si nKVDPCMOrder, que es uno de la pluralidad de indicadores se fija a 0), los primeros datos de valores de claves cuantizados se incluyen en la información de cabecera como datos de valores de intra claves. Si un valor de cuaternio de los primeros datos de valores de claves cuantizados en inverso satisface $\hat{Q}_0 = (\hat{q}_{0,0}, \hat{q}_{0,1}, \hat{q}_{0,2}, \hat{q}_{0,3})^T$, los primeros datos de valores de claves cuantizados en inverso se calculan con la ecuación (48) inferior.

$$\hat{q}_{0,0} = \tan\left(\frac{\pi}{4} * \left(\frac{|firstQKV\_S|}{2^{nKVQBit-1}-1}\right)\right) \qquad \cdots(48)$$

$$\hat{q}_{0,1} = \tan\left(\frac{\pi}{4} * \left(xSign * \frac{|firstQKV\_X|}{2^{nKVQBit-1}-1}\right)\right)$$

$$\hat{q}_{0,2} = \tan\left(\frac{\pi}{4} * \left(ySign * \frac{|firstQKV\_Y|}{2^{nKVQBit-1}-1}\right)\right)$$

$$\hat{q}_{0,3} = \tan\left(\frac{\pi}{4} * \left(zSign * \frac{|firstQKV\_Z|}{2^{nKVQBit-1}-1}\right)\right)$$

En la ecuación 48, xSign es 1 sólo cuando nFirsXSign en la clase OrilDPCMKeyValueHeader, que se describirá más adelante es 1, y es -1 en otras condiciones. ySign y zSing tienen la misma relación con nFirsYSign y nFirsZSign respectivamente, que la relación entre xSing y nFirsXSign

Los valores de las componentes de cuaternios restaurados definidos por la ecuación (48) se convierten en desplazamientos angulares a usar como un interpolador de orientación. Cada uno de los desplazamientos angulares restaurado a partir de los datos de valores de claves puede expresarse por un vector de cuatro dimensiones $(\hat{x}_i, \hat{y}_i, Z_i, \theta_i)^T$ donde i representa los datos de claves actuales, $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ representa el vector del eje de rotación, y $\hat{\theta}_i$ representa un ángulo de rotación en el sentido contrario de las agujas de reloj. Por consiguiente, los valores de las componentes de cuaternios restaurados se convierten en desplazamientos angulares siguiendo la ecuación (49).

$$\hat{x}_0 = \hat{q}_{0,1} * \frac{1}{\text{sen}(\frac{\hat{\theta}_0}{2})} \qquad \cdots(49)$$

$$\hat{y}_0 = \hat{q}_{0,2} * \frac{1}{\text{sen}(\frac{\hat{\theta}_0}{2})}$$

$$\hat{z}_0 = \hat{q}_{0,3} * \frac{1}{\text{sen}(\frac{\hat{\theta}_0}{2})}$$

$$\hat{\theta}_0 = \arccos(\hat{q}_{0,0}) * 2$$

El orden de la DPCM es 2, por ejemplo, si nKVDPCMOrder se fija a 1, los primeros y los segundos datos de valores de claves cuantizados se incluyen en la información de cabecera. Los primeros datos de valores de claves cuantizados se restauran del mismo modo que se ha descrito anteriormente. Los segundos datos de valores de claves, sin embargo, se restauran en un procedimiento diferente. En otras palabras, sólo tres componentes de los segundos datos de claves cuantizados se transmiten junto con un flujo de bits codificado y sus valores no son datos de valores de intra claves sino valores diferenciales ($\tilde{Q} = (\tilde{q}_{1,1}, \tilde{q}_{1,2}, \tilde{q}_{1,3})$) con los primeros datos de valores de claves. Suponiendo un cuaternio que representa los segundos datos de valores de claves de los datos de valores de claves cuantizados en inverso satisfacen $\hat{\tilde{Q}}_1 = (\hat{\tilde{q}}_{1,0}, \hat{\tilde{q}}_{1,1}, \hat{\tilde{q}}_{1,2}, \hat{\tilde{q}}_{1,3})^T$, $\hat{\tilde{Q}}_1$ se calcula con la ecuación (50) a continuación.

$$\hat{\tilde{q}}_{1,0} = \sqrt{1 - (\hat{\tilde{q}}_{1,1}^2 + \hat{\tilde{q}}_{1,2}^2 + \hat{\tilde{q}}_{1,3}^2)} \qquad \cdots(50)$$

$$\hat{\tilde{q}}_{1,1} = \tan(\frac{\pi}{4} * \left(\text{secondXSign} * \frac{|\text{secondQKV\_X}|}{2^{nKVQBit-1} - 1} \right)) \frac{V\_X|}{-1})$$

$$\hat{\tilde{q}}_{1,2} = \tan(\frac{\pi}{4} * \left(\text{secondYSign} * \frac{|\text{secondQKV\_Y}|}{2^{nKVQBit-1} - 1} \right)) \frac{V\_Y|}{-1})$$

$$\hat{\tilde{q}}_{1,3} = \tan(\frac{\pi}{4} * \left(\text{secondZSign} * \frac{|\text{secondQKV\_Z}|}{2^{nKVQBit-1} - 1} \right))$$

$$\hat{\tilde{q}}_{1,3} = \text{tang}(\frac{\pi}{4} * \left(\text{secondZSign} * \frac{|\text{secondQKV\_Z}|}{2^{nKVQBit-1} - 1} \right))$$

En la ecuación (50), secondXSign es 1 sólo cuando nSecondXSign en la clase OrilDPCMKeyValueHeader es 1, y es – 1 en otras condiciones, secondYSign y secondZSing tienen la misma relación con nSecondYSign y nSecondZSign respectivamente, que la relación entre secondXSing y nSecondXSign. Si un cuaternio $\hat{Q}_1$ que representa los segundos datos de valores de claves cuantizados inversos $\hat{Q}_1 = (\hat{q}_{1,0}, \hat{q}_{1,1}, \hat{q}_{1,2}, \hat{q}_{1,3})^T$, $\hat{Q}_1$ se calcula multiplicando $\hat{\tilde{Q}}_1$ por $\hat{Q}_0$. en otras palabras, $\hat{Q}_1 = \hat{\tilde{Q}}_1 \times \hat{Q}_0$.

El decodificador de cabecera 2110 saca los datos de valores de claves decodificados y la información de cabecera decodificada al decodificador de valores de claves 2150.

El decodificador de entropía 2410 recibe un flujo de bits, dentro del cual se codifican los datos diferenciales de los datos de valores de claves, y se decodifica en entropía el flujo de bits de la entrada usando la información de decodificación decodificada por el decodificador de cabecera 2110 en las etapas de S24120 hasta S24128.

La FIG. 25 es un diagrama que ilustra la estructura de una entrada de flujo de bits dentro del decodificador de entropía 2410. En la FIG. 25, suponiendo que N (nNumOfKeyValue) representa el número de datos de valores de claves codificados, el número de datos diferenciales rotacionales de cada una de las componentes incluidas en un flujo de bits es N -1 (0, 1, …, nNumberOfKeyValue – 2) cuando el orden de la DPCM es 0. Cuando el orden de la DPCM es 1, el número de datos diferenciales rotacionales de cada una de las componentes incluidas en el flujo de bits es N-2 (0, 1, …, nNumberOfKeyValue – 3).

El decodificador de entropía 2410 recibe, x_keyvalue_flag, y_keyvalue_flag, y z_keyvalue_flag desde el decodificador de cabecera 2110 y comprueba si x_keyvalue_flag, y_keyvalue_flag, y z_keyvalue_flag están todos fijados a 1 en la etapa S24120.

Cuando x_keyvalue_flag, y_keyvalue_flag, y z_keyvalue_flag están todos puestos a 0 se considera que todos los datos de valores de claves cuantizados o todos los datos diferenciales de cada uno de los componentes son iguales a nAllKeyValues en la clase OrilKeyValueCodingBit. Por consiguiente el decodificador de entropía 2410 decodifica los datos de valores de claves de cada uno de los componentes en el mismo valor que nAllKeyValues introducido desde el decodificador de cabecera 2110 y saca los datos de valores de claves decodificados para el operador de la DPCM circular inversa 2420 en la etapa S24122.

Si x_keyvalue_flag, y_keyvalue_flag, y z_keyvalue_flag no se fijan a cero, por ejemplo, si x_keyvalue_flag, y_keyvalue_flag, y z_keyvalue_flag se fijan a 1, el decodificador de entropía 2410 comprueba la entrada del modo de decodificación de entropía desde el decodificador de cabecera 2110 para decodificar cada una de las componentes de los datos de valores de claves de entrada $(\overline{Q}_i = (\overline{q}_{i,1}, \overline{q}_{i,2}, \overline{q}_{i,3}))$ en la etapa S24124.

Cuando el modo de decodificación de entropía es un modo de decodificación binaria, el decodificador de entropía 2410 decodifica un flujo de bits codificado de forma aritmética adaptativa usando una función decodeSignedAAC () mostrada en la Tabla 3 siguiente y saca los datos diferenciales rotacionales cuantizados al operador de la DPCM circular inversa 2420 en la etapa S24126.

Tabla 3

```
nulo decodeSignedAAC (entero *nDecodedValue, ent qstep, QState *signContext, Qstate
*valueContext) {
        entero b;
        b = qstep – 2;
        entero msb = 0;
        hacer {
                qf_decode(&msb, &valueContext [b]);
                msb =msb << b;
                b-;
        } mientras (msb == 0 && b>= 0);
        entero sgn = 0;
        entero rest = 0;
        si (msb! = 0) {
                qf_decode (&sgn, signContext);
                mientras (b >= 0) {
                        entero temp = 0;
                        qf_decode (&temp, &valueContext [b]);
                        rest |= (temp << b);
                        b--;
                }
        }
        si (sgn)
                *nDecodedValue = - (msb + rest);
        si no
                *nDecodedValue = (msb + rest);

  }
```

Por el contrario, cuando el modo de decodificación de entropía no es un modo de decodificación binaria, el decodificador de entropía 2410 decodifica el flujo de bits usando una función decodeUnaryAAC () en la etapa

S24128. La función decodeUnaryAAC () decodifica el flujo de bits de entrada leyendo consecutivamente 0 hasta que se lee 1 en el flujo de bits, convirtiendo el número de ceros sucesivos en su valor absoluto, y leyendo el siguiente bit a '1', y convirtiendo el bit en un bit de signo y a continuación saca los datos diferenciales rotacionales cuantizados al operador de la DPCM circular inversa 2420. Un ejemplo de la función decodeUnaryAAC se muestra en la Tabla 4

Tabla 4

```
Nulo decodeUnaryAAC (entero *nDecodedValue, QState* signContext, QState* valueContext)
{
        entero nBits = -1;
        bit bBit;
        hacer {
                qf_decode (&bit, valueContext);
                nBits++;
        } mientras (bBit == 0);
        si (nBits ! =0) {
                qf_decode (&bBit, signContext);
                si (bBit == 0)
                        *nDecodedValue = nBits;
                si no
                        *nDecodedValue = - nBits;
        }
        si no
                *nDecodedValue = 0;
 }
```

La función qf_decode () adoptada en las funciones mencionadas anteriormente decodeSignedAAC () y decodeUnaryAAc () se usa para leer el bit 1 de un flujo de bits codificados de forma aritmética adaptativa y se define por el documento ISO/IEC de 14496-2: 1999 Codificación de Objetos Audio Visuales: Visual, autorizado por la Organización de Normalización Internacional.

El operador de DPCM circular inversa 2420 recibe los datos de valores de claves decodificados en entropía desde el decodificador de entropía 2410 y comprueba el orden de la entrada de DPCM desde el decodificador de cabecera 2110. Si el orden de la DPCM es 0, el operador de la DPCM circular inversa 2420 saca los datos diferenciales rotacionales decodificados

$$\breve{Q}_{i-2}$$

al dispositivo de cuantización inversa 2430 en la etapa S24130 ya que la entrada de datos de valores de claves decodificadas en entropía procedentes del decodificador de entropía 2410 son datos diferenciales rotacionales cuantizados.

Por el contrario, si el orden de la DPCM es 1, el operador de la DPCM circular inversa 2420 realiza una operación de DPCM circular inversa en la etapa S24135 ya que la entrada de datos de valores de claves decodificados en entropía del decodificador de entropía 2410 son datos diferenciales rotacionales con la DPCM circular.

Suponiendo que nKVQBit representa los bits de cuantización inversa, el operador de DPCM circular inversa 2420 realiza una operación de DPCM circular inversa sobre los datos diferenciales rotacionales

$$\breve{Q}_{i-2}$$

siguiendo la ecuación (51) y genera los datos diferenciales rotacionales cuantizados

$$\breve{Q}'_{i-2}$$

en la etapa S24135

$$\breve{Q}'_{i-2} = \breve{Q}_{i-2} - (2^{nKVQBit} - 1) \quad (si\ \breve{Q}_{i-2} \geq 0)$$
$$\breve{Q}'_{i-2} = \breve{Q}_{i-2} + (2^{nKVQBit} - 1) \quad (si\ \breve{Q}_{i-2} \geq 0) \qquad \ldots(51)$$

(*i* = 2, .., *nNumberOfKeyValue* – 1)

En lo sucesivo, el operador de la DPCM circular inversa 2420 obtiene un valor A con la DPCM inversa y un valor B con la DPCM circular inversa, usando

$$\breve{Q}_{i-2}$$

y

$$\breve{Q}'_{i-2},$$

respectivamente, lo que se muestra en la ecuación (52).

$$A = \breve{Q}_{i-2} + \tilde{Q}_{i-1} \qquad \ldots(52)$$
$$B = \breve{Q}'_{i-2} + \tilde{Q}_{i-1}$$

(*i* = 2, .., *nNumberOfKeyValue* – 1)

El operador de la DPCM circular inversa 2420 saca B como datos con la DPCM circular inversa $\tilde{Q}_i$ si B + ($2^{nKVQBit\ -1}$ - 1) está entre 0 y un valor máximo en un intervalo de cuantización. Por el contrario, el operador de la DPCM circular inversa 2420 saca A como $\tilde{Q}_i$ si B + ($2^{nKVQBit\ -1}$ -1) es menor que 0 o mayor que el valor máximo en el intervalo de cuantización.

Un ejemplo de códigos de programa C++, en el cual está escrita la operación mencionada anteriormente del operador de la DPCM circular inversa 2420, se muestra en la Tabla 5.

Tabla 5

```
ICDPCM (entero *curIDPCMKeyValue, entero deltaValue, entero prevICDPCMKeyValue)
{
        entero circularDelta;
        entero tempIDPCMKeyValue;
        prevICDPCMKeyValue + = ((1 << (nKVQBit – 1)) – 1);

        si (deltaKeyValue >= 0,0)
                circularDelta = deltaKeyValue – ((1 << nKVBit) – 1);
        si no
                circularDelta = deltaKeyValue + ((1 << nKVBit) – 1);

        tempIDPCMKeyValue = circularDelta + prevICDPCMKeyValue;

        si ((temIDPCMKeyValue >= 0,0) && (tempIDPCMKeyValue < ((1 << nKVQBit) – 1)))
                *curIDPCMKeyValue = tempIDPCMKeyValue;
        si no
                *curIDPCMKeyValue = deltaKeyValue + prevICDPCMKeyValue;

        *curIDPCMKeyValue - = ((1 << (nKVQBit – 1)) – 1);
 }
```

El dispositivo de cuantización inversa 2430 recibe los datos diferenciales rotacionales cuantizados ($\tilde{Q}_i = (\tilde{q}_{i,0}, \tilde{q}_{i,1}, \tilde{q}_{i,2}, \tilde{q}_{i,3})$) generados por el operador de la DPCM circular inversa 2420 realizando una operación de DPCM circular inversa, restaura un valor diferencial rotacional $(\hat{\tilde{Q}}_i = (\hat{\tilde{q}}_{i,0}, \hat{\tilde{q}}_{i,1}, \hat{\tilde{q}}_{i,2}, \hat{\tilde{q}}_{i,3})^T)$ realizando una operación de cuantización inversa sobre $\tilde{Q}i$ siguiendo la ecuación (53), y saca el valor diferencial rotacional $\hat{\tilde{Q}}_i$ al multiplicador de cuaternios 2440 en la etapa S24140.

$$\hat{\tilde{q}}_{i,0} = \sqrt{1-(\hat{\tilde{q}}_{i,1}^2 + \hat{\tilde{q}}_{i,2}^2 + \hat{\tilde{q}}_{i,3}^2)} \qquad \ldots(53)$$

$$\hat{\tilde{q}}_{i,j} = \text{tang}\left(\frac{\pi}{4} * \left(\text{sgn}(\tilde{q}_{i,j}) * \frac{|\tilde{q}_{i,j}|}{2^{nKVQBit-1}-1}\right)\right)$$

$$(i = 2,\ldots, nNumberOfKeyValue - 1, \quad j = 1,2,3)$$

En lo sucesivo, el multiplicador de cuaternios 2440 recibe un valor diferencial rotacional $\hat{\tilde{Q}}_i$. El multiplicador de Cuaternios 2440 restaura el valor de transformación rotacional $\hat{Q}i$ (donde $\hat{Q}i = (\hat{q}_{i,0}, \hat{q}_{i,1}, \hat{q}_{i,2}, \hat{q}_{i,3})^T$) de un fotograma clave actual en la etapa S24154 por la multiplicación de cuaternios del valor diferencial rotacional de entrada $\hat{\tilde{Q}}_i$ por el valor de la transformación rotacional $\hat{Q}_{i-1}$ de un fotograma clave anterior siguiendo la ecuación (54).

$$\hat{Q}_i = \hat{\tilde{Q}}_i \times \hat{Q}_{i-1} \qquad \ldots(54)$$

(*i* = 2, .., *nNumberOfKeyValue* – 1)

Después de restaurar el valor de la transformación rotacional, el decodificador de datos de valores de claves 2400 comprueba si los valores de transformación rotacional restaurados corresponden a los últimos datos de valores de claves en la etapa S24156. Si el valor de la transformación rotacional restaurada no corresponde a los últimos datos de valores de claves, el decodificador de datos de valores de claves 2400 realiza repetidamente las etapas S24140 hasta S24154. Por el contrario, si los valores de la transformación rotacional restaurados son los últimos datos de valores de claves, el decodificador de los datos de valores de claves 2400 saca el valor de la transformación rotacional restaurado en la etapa S24158.

Si el orden de la DPCM es 0, el operador de la DPCM circular inversa 2420 saca los datos diferenciales rotacionales cuantizados decodificados

$$\breve{Q}_i$$

(donde

$$\breve{Q}_i = (\breve{q}_{i,1}, \breve{q}_{i,2}, \breve{q}_{i,3})$$

al dispositivo de cuantización inversa 2430. A continuación el dispositivo de cuantización inversa 2430 genera un valor diferencial rotacional $\hat{\tilde{Q}}_i$ (donde $\hat{\tilde{Q}}_i = (\hat{\tilde{q}}_{i,0}, \hat{\tilde{q}}_{i,1}, \hat{\tilde{q}}_{i,2}, \hat{\tilde{q}}_{i,3})^T$ por la cuantización inversa de los datos diferenciales rotacionales

$$\breve{Q}_i$$

introducidos desde el operador de DPCM circular 2420 y saca $\hat{\tilde{Q}}_i$ al multiplicador de cuaternios 2440 en la etapa S24140.

$$\hat{\tilde{q}}_{i,0} = \sqrt{1-(\hat{\tilde{q}}_{i,1}^2 + \hat{\tilde{q}}_{i,2}^2 + \hat{\tilde{q}}_{i,3}^2)} \qquad \ldots(55)$$

$$\hat{\tilde{q}}_{i,j} = \text{tang}\left(\frac{\pi}{4} * \left(\text{sgn}(\tilde{q}_{i,j}) * \frac{|\tilde{q}_{i,j}|}{2^{nKVQBit-1}-1}\right)\right)$$

$$(i = 1,\ldots, nNumberOfKeyValue - 1, \quad j = 1,2,3)$$

Incluso cuando el orden de la DPCM es 0, el multiplicador de cuaternios 2440 restaura los valores de la transformación rotacional casi del mismo modo (las etapas S24250 hasta S24158) que lo hace cuando el orden de la DPCM no es 0. Cuando el orden de la DPCM es 0, el multiplicador de cuaternios 2440 realiza la multiplicación de cuaternios siguiendo la ecuación (56)

$$\hat{Q}_i = \tilde{\hat{Q}}_i \times \hat{Q}_{i-1} \quad \ldots(56)$$

(*i* = 1, .., *nNumberOfKeyValue* – 1)

En lo sucesivo, se describirá la operación del sintetizador del interpolador de orientación 2180.

La FIG. 26 es un diagrama de flujo de la operación del sintetizador del interpolador de orientación 2180. Refiriéndonos a la FIG. 26, el sintetizador del interpolador de orientación 2180 recibe los datos de claves decodificados y los datos de valores de claves decodificados y recibe el número de datos de claves y el modo de generación de puntos de ruptura y los indicadores de selección de claves desde el decodificador de cabecera 2110 en la etapa S26000.

El sintetizador del interpolador de orientación 2180 saca los datos de claves decodificados y los datos de valores de claves decodificados en la etapa S26100 cuando el modo de generación de puntos de ruptura es un modo de conservación de la trayectoria de animación. En el caso de que los puntos de ruptura se hayan extraído en un modo de conservación de la trayectoria de animación, los datos de claves corresponden a los datos de valores de claves, y de este modo no es necesario generar nuevos datos de valores de claves usando la interpolación. Sin embargo, si los puntos de ruptura se han extraído en un modo de conservación de las claves de animación, esto significa que sólo los datos de valores de claves correspondientes a los puntos de ruptura se han codificado mientras que todos los datos de claves se han codificado y decodificado. Por consiguiente, en esta caso, los datos de claves no corresponden a los datos de valores de claves, y de ese modo es necesario interpolar los datos de valores de claves, que se describirán en los siguientes párrafos.

Cuando el modo de generación de puntos de ruptura es un modo de conservación de las claves de animación, un contador, que indica un índice de la disposición de indicadores de selección de claves, se establece de modo que corresponde a los datos de claves en la etapa S26200, para descifrar si existen o no los datos de valores de claves correspondientes a los datos de claves.

El sintetizador del interpolador de orientación 2180 comprueba si los datos de valores de claves correspondientes a los datos de claves indicados por el contador existen y va a la etapa S26500 en la etapa S26300.

Si no hay ningún dato de valor de clave correspondiente a los datos de claves indicados por el contador, el sintetizador del interpolador de orientación 2180 genera los datos de valores de claves en la etapa S26400 por la interpolación lineal de los datos de valores de claves actuales usando los datos de valores de claves correspondientes a los datos de claves indicados por dos puntos de ruptura sucesivos incluyendo uno anterior a los datos de claves indicados por el contador y otro próximo a los datos de claves indicados por el contador.

Después de esto, el sintetizador del interpolador de orientación 2180 comprueba si todos los datos de valores de claves tienen sus datos de valores de claves correspondientes y se ha realizado la interpolación lineal sobre todos los datos de claves que no tienen sus datos de valores de claves correspondientes en la etapa S26500. Si aún existen datos de claves, que no se han comprobado, el sintetizador del interpolador de orientación 2180 renueva el contador y realiza las etapas de S26300 hasta S26500 de nuevo. Si se han comprobado todos los datos de claves, el sintetizador del interpolador de orientación 2180 saca los datos de claves sintetizados y los datos de valores de claves sintetizados como un interpolador de orientación en la etapa S26600.

En lo sucesivo, se describirán ejemplos de códigos de programa de lenguaje SDL, por los cuales se realiza el aparato para la decodificación de un interpolador lineal de acuerdo con la presente invención, que decodifica datos de claves y datos de valores de claves.

La FIG 29A es un diagrama que ilustra un clase de Interpolador de Orientación Comprimido (CompressedOrientationInterpolator). El Interpolador de Orientación Comprimido es una clase superior usada para la lectura de un flujo de bits codificado de un interpolador de orientación, La cabecera de claves (KeyHeader, el Indicador de selección de claves (KeySelectionFlag), y la clave (Key) son clases para la lectura desde un flujo de bits, de la información de los datos de claves correspondientes a los datos del campo de claves en una orientación convencional. OrilKeyValueHeader y OrilDPCMKeyValue son clases para la información de lectura sobre los datos de valores de claves correspondientes a los datos del campo de valores de claves en el interpolador de orientación convencional. Se usa una función qf_start () para inicializar un decodificador aritmético antes de la lectura de un flujo de bits, dentro del cual se codifican los datos de valores de claves usando la codificación aritmética adaptativa (AAC).

La FIG. 29B es un diagrama que ilustra una clase de cabecera de claves para la decodificación de una cabecera de claves requerida para la decodificación.

La clase de cabecera de claves incluye el número de datos de claves, los bits de cuantización, los datos de intra claves, una cabecera de DND y el número de bits realmente usados para la decodificación. nKeyBit representa los bits de cuantización usados en la cuantización inversa para restaurar los valores de las claves de punto flotante. nNumKeyCodingBit indica un tamaño de bits de nNumberOfKey que representa el número de datos de claves. nKeyDigit indica un número de dígitos más significativos de los datos de claves originales y se usa para redondear los valores decodificados.

Cuando se incluye la información sobre las sub-regiones de claves lineales en una cabecera de claves, el indicador bIsLinearKeySubRegion se pone a 1. En este caso, las claves incluidas en sub-regiones lineales de claves específicas dentro de una clave entera, puede calcularse la región usando la información de cabecera decodificada siguiendo el indicador bIsLinearKeySubRegion. bRangeFlag indica si los datos de claves varían o no desde 0 hasta 1. Si los datos de claves no varían desde 0 hasta 1, los valores mínimo y máximo dentro del intervalo de datos de claves se decodifican desde la clase KeyMinMax. KeyMinMax recupera los valores mínimo y máximo requeridos para la cuantización inversa. Cada uno de los valores mencionados anteriormente puede dividirse en sus mantisas y exponentes respectivos.

nBitSize es el tamaño de bits de nQIntrakey, y nKeyShift es un tamaño de bits inicial de nKeyMax, nQIntraKey indica la magnitud de los primeros intra datos cuantizados y se combina con nQIntraKeySign que indica el signo de nQIntraKey, nQIntraKey se usa como una base requerida para la restauración de otos datos de claves cuantizados. En todos los bits de signo usados en la compresión del interpolador, un valor de '0' indica un signo positivo, y un valor de '1' indica un signo negativo. nKDPCMOrder indica un valor, que es el orden de la DPCM menos 1. El orden de la DPCM puede ser 1, 2 ó 3. El número de intra datos cuantizados es el mismo que el orden de la DPCM.

nKeyShift, junto con el bit de signo nKeyShiftSign, es un número entero que indica la cantidad de desplazamiento en un decodificador de datos de claves. nKeyShift y nKeyShiftSign se decodifican si bShiftFlag está puesto a 'verdadero'. nDNDOrder es el orden de la operación de división y división (DND). La operación de DND se ha descrito anteriormente con un decodificador de datos de claves. Si el valor de nNDNOrder es 7, entonces bNoDND se decodifica. bNoDND indica si se realizará o no una operación inversa de DND. nKeyMax es un valor máximo o un valor mínimo usado durante los ciclos sucesivos de una operación de DND inversa. nKeyCodingBit indica los bits usados para la codificación de los datos de claves.

bSignedAACFlag indica un procedimiento de decodificación a realizar durante la decodificación de AAC. Si bSignedAACFlag se fija a 0, se realizará un procedimiento de decodificación AAC sin signo. De lo contrario, se realizará un procedimiento de decodificación de AAC con signo. bKeyInvertDownFlag es un valor Booleano que indica si se usará o no nKeyInvertDown. nKeyInvertDown es un número entero para convertir todos los datos de claves cuantizados de valores negativos mayores que el mismo en valores negativos no mayores que -1. Si nKeyInvertDown se fija a -1, entonces no se realizará una operación de desplazamiento hacia abajo.

La FIG. 29C es un diagrama que ilustra una clase de LinearKey. En la FIG. 29C, nNumLinearkeyCodingBit es un valor que indica el número de bits necesario para codificar un número predeterminado de claves predecibles linealmente. nNumberOfLinearKey es un valor que indica el número de claves predecibles linealmente.

La FIG. 29D es un diagrama que ilustra la clase keyMinMax. En la FIG. 29D, bMinKeyDigitSame es una bandera que indica si el número (nKeyDigit) de los dígitos más significativos de todas las claves y el número de dígitos más significativos de un valor mínimo de entre las claves son los mismos. nMinKeyDigit es un valor que indica el número de dígitos más significativos del valor mínimo de entre las claves. nMinKeyMantissaSign es un valor que indica el signo de nMinKeyMantissa. nMinKeyMantissa es un valor que indica la mantisa del valor mínimo de entre las claves. nMinKeyExponentSign es un valor que indica el signo de nMinKeyExponent.

nMinkeyExponent es un valor que indica el exponente del valor mínimo de entre las claves.

fKeyMin es un valor que indica el valor mínimo de entre las claves. bMaxKeyDigitSame es una bandera que indica si el número nKeyDigit de los dígitos más significativos de todas las claves y el número de dígitos más significativos de un valor máximo de entre las claves son los mismos, nMaxKeyDigit es un valor que indica el número de dígitos más significativos del valor máximo de entre las claves. nMaxKeyMantissaSign es un valor que indica el signo de nMaxKeyMantissa. nMaxKeyMantissa es un valor que indica la mantisa del valor máximo de entre las claves.

bSameExponent es una bandera que indica si el exponente del valor máximo de entre las claves es el mismo que nMinKeyExponent. nMaxKeyExponentSign es un valor que indica el signo de nMaxKeyExponent. nMaxKeyExponent es un valor que indica el exponente del valor máximo de entre las claves. fKeyMax es un valor que indica el valor máximo de entre las claves.

La FIG. 29E es un diagrama que ilustra una clase de OrilKeyValueHeader. El significado de cada una de las variables usadas en esta clase es como sigue. bPreserveKey indica si el modo de decodificación actual es un modo de conservación de la clave de animación o un modo de conservación de la trayectoria de animación. Cuando bPreserveKey se fija a "cierto", el modo de decodificación actual es un modo de conservación de claves de animación.

nKVQBit representa un tamaño de bits de cuantización inversa de los datos de valores de claves, nKVDPCMOrder representa el orden de la operación de DPCM inversa usada para la decodificación de datos de valores de claves. Cuando nKVDPCMOrder = 0, no necesita realzarse la operación de DPCM circular inversa. Por el contrario, cuando nKVDPCMOrder = 1, se supone que se va a realizar una operación de DPCM circular inversa que tiene un orden de 2.

La FIG. 29F es un diagrama que ilustra una clase de OriDPCMKeyValueHeader. El significado de cada una de las variables usadas en esta clase es como sigue. firstQKV_S, firstQKV_X, firstQKV_Y y firstQKV_Z representan los primeros valores de las cuatro componentes s, x, y, y z, respectivamente, que constituyen un cuaternio (s, x, y, z) que representa datos de valores de claves cuantizados. nFirstXSign, nFirstYSign y nFirstZSign representan los signos de firstQKV_X, firstQKV_Y y firstQKV_Z respectivamente. secondQKV_X, secondQKV_Y y secondQKV_Z, representan los segundos valores de las tres componentes x, y, y z, respectivamente y nSecondXSign, nSecondYSign, y nSecondZSign representan los signos de secondQKV_X, secondQKV_Y y secondQKV_Z. respectivamente. bisMoreTwoKVs indica si hay más de dos datos de valores de claves a decodificar cuando se supone que se va a realizar una DPCM circular inversa. x_keyvalue_flag, y_keyvalue_flag, y z_keyvalue_flag indican si todos los valores cuantizados de cada una de las componentes x, y , y z, respectivamente, son los mismos.

La FIG. 29G es un diagrama que ilustra una clase de OriIKeyValueCodingBit. El significado de cada una de las variables usadas en esta clase es como sigue. nKVCodingBit representa el número de bits usados para almacenar cada una de las componentes de todos los datos de valores de claves de cuaternios excepto para los datos de valores de intra claves (firstQKV_S, FirstQKV_X, FirstQKV_Y, FirstQKV_Z, secondQKV_X, secondQKV_Y y secondQKV_Z en la clase OriIDPCMKeyValueHeader) después de la cuantización.

nAllKeyValue representa un valor cuantizado de cada una de las componentes de todos los datos de valores de claves cuando keyvalue_flag para cada una de las componentes se fija a 0. nSing representa el signo de nAllKeyValue. bIsUnaryAAC representa un procedimiento de cuantización aritmética adaptativa usado para la decodificación de valores cuantizados de cada uno de las componentes x, y, y z. Si bIsUnaryAAC se fija a 'cierto' se usará una función de decodificación AAC unaria. Por el contrario, si bIsUnaryAAC se fija a 'falso' se usará una función de decodificación AAC binaria.

La FIG. 29H es un diagrama que ilustra una clase de KeySelectionFlag. En la clase KeySelectionFlag, KeyFlag indica si se ha codificado o no el dato de clave de orden i. nNumOfKeyValue es un número entero que indica el número de datos de valores de claves a decodificar.

La FIG. 29I es un diagrama que ilustra una clave de clase. En la clave de clase Key, nQKey es una disposición de datos de claves cuantizados decodificados desde un flujo de bits. KeyContext es un contexto para la lectura de la magnitud de nQKey. KeySignContext es un contexto para la lectura del signo de nQkey.

DecodeUnsignedAAC es una función usada para realizar un procedimiento de decodificación sin signo de la codificación aritmética adaptativa con un contexto determinado, que se describirá más adelante. DecodeSignedAAC es una función usada para realizar un procedimiento de decodificación con signo de la codificación aritmética adaptativa con un contexto determinado que se describirá a continuación,

La FIG. 29J es un diagrama que ilustra una clase de OriIDPCMKeyVaule. El significado de cada uno de los valores usados en la clase es como sigue. DeltaKeyValue se usa para almacenar datos de valores de claves cuantizados incluyendo las tres componentes x, y y z en la forma de cuaternios. Los datos de valores de claves cuantizados almacenados en DeltaKeyValue se decodifican a partir del flujo de bits usando la función decodeUnaryAAC o decodeSignedAAC.

kVXSignContext, kVYSignContext, y kVYZSignContext son contextos usados para la decodificación de las tres componentes x, y, y z de DeltaKeyValue usando la función decodeUnaryAAC o decodeSignedAAC.

kVXUnaryContext, kVYUnaryContext y kVZUnaryContext son contextos usados para la decodificación de las tres componentes x, y, y z de DeltaKeyValue usando la función decodeUnaryAAC.

kVXContext, kVYContext, y kVZContext son los contextos usados para la decodificación de las tres componentes x, y, y z de DeltaKeyValue usando la función decodeSignedAAC.

En lo sucesivo, se describirá un procedimiento de medición de un error entre el valor diferencial rotacional original y un valor diferencial rotacional restaurado por la cuantización inversa de un valor diferencial rotacional cuantizado en el calculador de errores 42b, 46b, y 48b incluidos en los extractores de puntos de ruptura 42, 46 y 48 mostrados en las FIG. 4A hasta 4C, respectivamente y un procedimiento de medida del error entre un valor diferencial rotacional original y un valor diferencial rotacional restaurado por la cuantización inversa de un valor diferencial rotacional cuantizado, que se realiza en la unidad de mediciones de error 2050 incluida en el dispositivo de cuantización 2000 de acuerdo con la tercera realización de la presente invención.

El procedimiento de medición de un error entre el valor diferencial rotacional original y el valor diferencial rotacional restaurado de acuerdo con la presente invención también puede usarse para medir un error entre un interpolador de

orientación original antes de la codificación y un interpolador de orientación generado por la decodificación de un interpolador de orientación codificado. Por consiguiente, se describirá una trayectoria de animación de un interpolador de orientación, desde la cual se extraerán los puntos de ruptura por los extractores de los puntos de ruptura 42, 46 ó 48, y el valor diferencial rotacional original usado en la unidad de mediciones de error 2050 como correspondiente a un interpolador de orientación antes de la codificación. Del mismo modo, se describirá una trayectoria de animación de un interpolador de orientación constituido por los puntos de ruptura extraídos y los valores diferenciales rotacionales restaurados usados en la unidad de mediciones de error 2050 como correspondientes a los datos de valores de claves del interpolador de orientación decodificado.

En el procedimiento de codificación de un interpolador de orientación, ocurre un error entre un interpolador de orientación original y un interpolador de orientación restaurado durante la cuantización. En este punto, el error entre el interpolador de orientación original y el interpolador de orientación restaurado se define por una diferencia en los ángulos entre la transformación rotacional original y la transformación rotacional restaurada.

En otras palabras, suponiendo que un dato de valor de clave incluido en un nodo interpolador de orientación y su dato de valor de clave restaurado en un decodificador se denominan como ($\vec{r}$,0) y ($\vec{r}$,$\hat{\theta}$), respectivamente, donde $\vec{r}$ representa un eje de rotación, y θ representa un ángulo de rotación y satisface θ ∈[- π, π] y que un objeto en un espacio tridimensional se mueve desde una posición arbitraria $\vec{x}$ a una posición arbitraria $\vec{y}$ y desde $\vec{y}$ una posición arbitraria $\hat{\vec{y}}$ en base a ($\vec{r}$, θ) y ($\vec{r}$, $\hat{\theta}$), respectivamente, por transformación de rotación, el error de cuantización es la diferencia entre $\vec{y}$ e $\hat{\vec{y}}$ y satisface $\vec{e}(\vec{x})$ = $\vec{y}$ - $\hat{\vec{y}}$, $\vec{x}$, $\vec{y}$, e $\hat{\vec{y}}$ representados en la forma de cuaternios se representan en la ecuación (57).

$$X = (0, \vec{x}), \quad Y = (0, \vec{y}), \quad \hat{Y} = (0, \hat{\vec{y}}) \qquad \ldots(33)$$

Donde los cuaternios que representan ($\vec{r}$, θ) y ($\vec{r}$, ^θ) se denominan como Q y $\hat{Q}$, pueden deducirse de las siguientes ecuaciones

$$Y = Q \times X \times Q^* \qquad \ldots(58)$$

$$X = Q^* \times Y \times Q$$

En este punto, A x B representa la multiplicación de cuaternios, y A* representa el conjugado de A. En base a las ecuaciones (57) y (58), puede deducirse la siguiente ecuación.

$$\hat{Y} = \hat{Q} \times X \times \hat{Q}^* = \hat{Q} \times Q^* \times Y \times Q \times \hat{Q}^* = Q_{error} \times Y \times Q^*_{error} \qquad \ldots(59)$$

En este punto, Q_{error} es un valor que representa la relación entre $\vec{y}$ y $\hat{\vec{y}}$ en términos de la transformación de rotación y se defina por la siguiente ecuación.

$$Q_{error} = \hat{Q} \times Q^* \qquad \ldots(60)$$

Cuando un ángulo de rotación diferencial entre $\vec{y}$ y $\hat{\vec{y}}$ se define como θ_{error}, θ_{error} puede obtenerse usando la fórmula de transformación de cuaternios y la ecuación (61).

$$\theta_{error} = 2\cos^{-1} q_{0,\,error} = 2\cos^{-1}(\hat{Q} \cdot Q) \qquad \theta_{error} \in [0, \pi] \qquad \ldots(61)$$

En este punto, * representa una operación de un producto interior. La ecuación (61) define un error de cuantización instantáneo que ocurre entre todos los fotogramas clave de animación en un momento de tiempo predeterminado, y por consiguiente, la unidad de medición de errores 2050 incluida en el dispositivo de cuantización 2000 de acuerdo con la tercera realización de la presente invención calcula el error entre un interpolador de orientación original y un interpolador de orientación restaurado con la ecuación (61).

Además, un error de cuantización instantáneo en un momento de tiempo predeterminado (t) puede definirse por la siguiente ecuación para deducir una fórmula para obtener un error de cuantización en todo el intervalo de animación a partir de la ecuación (62)

$$e(t) = 2\arccos\{Q(t) \cdot \hat{Q}(t)\} \qquad \ldots(62)$$

El error promedio E_m y el error máximo E_p pueden deducirse aplicando extensivamente la ecuación (62) a todo el intervalo del fotograma clave, durante el cual se está realizando la animación usando un interpolador de orientación.

En este punto, para obtener el error promedio E_m, la suma parcial E^i_m de los errores en el intervalo $[\hat{t}_{j-1}, \hat{t}_j]$ debe obtenerse en primer lugar, como se muestra en la FIG. 27. En lo sucesivo, los momentos predeterminados de

tiempo en los que los datos de claves decodificados correspondientes a los datos de claves originales en los momentos de tiempo predeterminados t_{i-1}, y t_i existentes se denominarán respectivamente como $\hat{t}_{i-1}$ y $\hat{t}_i$, y los datos de valores de claves decodificados correspondientes a los datos de valores de claves originales Q_{i-1} y Q_i se denominan $\hat{Q}_{i-1}$ y $\hat{Q}_i$ respectivamente.

Debido al ruido generado durante la codificación de los datos de claves, es imposible calcular directamente un error entre una trayectoria de animación original Q_i' de un interpolador de orientación original y una trayectoria de animación $\hat{Q}_i$ 'de un interpolador de orientación decodificado, como se muestra en la FIG. 27. Por consiguiente, el intervalo $[t_{i-1}, t_i]$ debe dividirse en tres sub-intervalos $[t_{i-1}, \hat{t}_{i-1}]$, $[\hat{t}_{i-1}, \hat{t}_i]$, y $[\hat{t}_i, t_i]$. A continuación, los valores de cuaternios en $\hat{t}_{i-1}$ y $\hat{t}_i$ se obtienen usando la ecuación (63).

$$Q'_{i-1} = SLERP(Q_{i-1}, Q_i, \frac{\hat{t}_{i-1} - t_{i-1}}{t_i - t_{i-1}})$$
$$Q'_i = SLERP(Q_{i-1}, Q_i, \frac{\hat{t}_i - t_{i-1}}{t_i - t_{i-1}}) \quad \ldots(63)$$

En la ecuación (63), se usa una función SLERP () para realizar la interpolación lineal esférica.

Debido al hecho de que una trayectoria de animación está inevitablemente distorsionada en un espacio de 3D, como se muestra en la FIG. 27, $[\hat{t}_{i-1}, \hat{t}_i]$ debe dividirse en dos sub-intervalos $[\hat{t}_{i-1}, t_i'']$ y $[t_i'', \hat{t}_i]$ y a continuación los dos sub-intervalos deben calcularse separadamente entre sí. En este punto asumamos que una distancia entre las dos trayectorias de animación, es decir entre un grupo de valores de Q_i y un grupo de valores de $\hat{Q}_i$ se minimiza en t_i'' en el intervalo $\{\hat{t}_{i-1}, \hat{t}_i]$. Los errores instantáneos en $\hat{t}_{i-1}$ y $\hat{t}_i$ se calculan usando las siguientes ecuaciones.

$$e_{i-1} = 2\arccos(Q'_{i-1} \cdot \hat{Q}_{i-1})$$
$$e_i = 2\arccos(Q'_i \cdot \hat{Q}_i) \quad \ldots(64)$$

Además, t_i'' en el intervalos $[\hat{t}_{i-1}, \hat{t}_i]$ es proporcional a $\frac{e_{i-1}}{e_i}$, que se muestra en la ecuación (65).

$$t_i'' = \hat{t}_{i-1} + \frac{e_{i-1}}{e_{i-1} + e_i}(\hat{t}_i - \hat{t}_{i-1}) \quad \ldots(65)$$

Los valores intermedios de cuaternios y el error instantáneo en ti'' se calculan con la ecuación (66) a continuación.

$$Q_i'' = \mathrm{SLERP}(Q'_{i-1}, Q'_i, \frac{t_i'' - \hat{t}_{i-1}}{\hat{t}_i - \hat{t}_{i-1}})$$
$$\hat{Q}_i'' = \mathrm{SLERP}(\hat{Q}_{i-1}, \hat{Q}_i, \frac{t_i'' - \hat{t}_{i-1}}{\hat{t}_i - \hat{t}_{i-1}}) \quad \ldots(66)$$
$$e_i'' = 2 \cdot \arccos(Q_i'' \bullet \hat{Q}_i'')$$

Sin embargo, un error instantáneo en un momento arbitrario (t) del tiempo se obtiene siguiendo la siguiente ecuación (67).

$$e(t) = 2 \cdot \arccos(Q(t) \cdot \hat{Q}(t)) \quad \ldots(67)$$

En este punto, $Q(t) = \mathrm{SLERP}(Q'_{i-1}, Q'_i, \frac{t - \hat{t}_{i-1}}{\hat{t}_i - \hat{t}_{i-1}})$ y $\hat{Q}(t) = SLERP(\hat{Q}_{i-1}, \hat{Q}_i, \frac{t - \hat{t}_{i-1}}{\hat{t}_i - \hat{t}_{i-1}})$.

Sin embargo, no es fácil calcular un error instantáneo e(t) en un momento arbitrario (t) del tiempo. Por consiguiente, e(t) se determina usando una aproximación lineal, que se muestra en la aproximación (68).

$$e(t) \cong \begin{cases} e_{i-1} + \frac{t - \hat{t}_{i-1}}{t_i'' - \hat{t}_{i-1}}(e_i'' - e_{i-1}) & , \quad \hat{t}_{i-1} \le t \le t_i'' \\ e_i'' + \frac{t - t_i''}{\hat{t}_i - t_i''}(e_i - e_i'') & , \quad t_i'' \le t \le \hat{t}_i \end{cases}$$

...(68)

La suma parcial E'_m de los errores en el intervalo [^t_{i-1}, ^t_i] y la suma parcial E'_p de los errores máximos en el intervalo [^t_{i-1}, ^t_i] pueden obtenerse también usando las aproximaciones (69) y (70).

$$\begin{cases} E_p^i \equiv \max_{\hat{t}_{i-1} \le t \le \hat{t}_i} |e(t)| \\ E_m^i \equiv \int_{t_{i-1}}^{t_i} e^2(t)dt + \int_{t_i}^{\hat{t}_i} e^2(t)dt \end{cases}, \quad \text{...(69)}$$

La aproximación de (69) puede re-disponerse como sigue:

$$\begin{cases} E_p^i \cong \max\{|e_{i-1}|, |e_i''|, |e_i|\} \\ E_m^i \cong \frac{1}{3}(t_i'' - \hat{t}_{i-1})\{(e_i'')^2 + e_{i-1}^2 + e_i'' e_{i-1}\} + \frac{1}{3}(\hat{t}_i - t_i'')\{e_i^2 + (e_i'')^2 + e_i e_i''\} \end{cases} \quad \text{..(70)}$$

Finalmente, se calculan un error promedio E_m y un error máximo E_p en un intervalo de tiempo [t_{min}, t_{max}] con la aproximación (71) a continuación.

$$E_m \cong \sqrt{\frac{1}{t_{max} - t_{min}} \sum_i E_m^i} \quad \text{...(71)}$$

$$E_p \cong \max_i E_p^i$$

Por consiguiente, el calculador de errores 42b, 46b y 48b incluidos en el extractor de puntos de ruptura 42, 46, y 48 mostrados en las FIG. 4A hasta 4C, respectivamente, calcula un error entre un interpolador de orientación original y un interpolador de orientación restaurado usando la ecuación (71) y puede medirse el grado en el que se distorsionan las imágenes en un espacio de cuaternios debido a los errores de cuantización de forma más precisa.

La FIG. 28 es un diagrama que muestra el funcionamiento del procedimiento para la codificación de datos de valores de claves de acuerdo con la presente invención en comparación con un procedimiento convencional para la codificación de datos de valores de claves. Como se muestra en la FIG. 28, de acuerdo con el procedimiento para la codificación de un interpolador de orientación de la presente invención, el grado en el que se distorsionan las imágenes cuando se da un número de bits predeterminado requerido para la codificación, puede ser considerablemente más bajo que en el procedimiento convencional MPEG-4 BIFS PMFC.

La presente invención puede realizarse como códigos legibles por ordenador escritos sobre un medio de grabación legible por ordenador. En este punto, el medio de grabación legible por ordenador incluye cualquier clase de medio de grabación que pueda leerse por un sistema de ordenador. Por ejemplo, el medio de grabación legible por ordenador puede incluir una ROM, una RAM, un CD-ROM, una cinta magnética, un disco flexible, un almacenamiento óptico de datos, una onda portadora (transmisión a través de la Internet) y similares. El medio de grabación legible por ordenador puede ser descentralizado para sistemas de ordenadores conectados sobre una red, y el ordenador puede leer el medio de grabación de un modo descentralizado.

El procedimiento y el aparato para la codificación de un interpolador de orientación de acuerdo con la presente invención son capaces de codificar los datos de claves de animación y los datos de valores de claves con una eficacia elevada mientras que se mantiene una animación de alta calidad. Además de acuerdo con la presente invención, es posible mantener datos de animación de alta calidad aprovechando un procedimiento de medición de un error entre las trayectorias de datos de animación, y también es posible reducir considerablemente la cantidad de datos de claves y datos de valores de claves a codificar usando el procedimiento de re-muestreo y el punto de extracción de los puntos de ruptura de la presente invención.

Además es posible proporcionar una animación de alta calidad con una alta tasa de compresión calculando un valor diferencial rotacional, que puede reflejar de forma suficiente la redundancia en la transformación rotacional correspondiente a los datos de valores de claves de un interpolador de orientación, y de este modo codificar los datos de valores de claves del interpolador de orientación.

Aunque la invención se ha mostrado y se ha descrito particularmente con referencia a las realizaciones preferidas de la misma, se entenderá por los expertos en la materia que pueden realizarse diversos cambios en la forma y en los detalles de las mismas sin apartarse del alcance de la invención como se define por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un aparato para la codificación de un interpolador de orientación usado en la animación de fotogramas claves de un objeto en 3D, incluyendo el interpolador de orientación datos de claves que indican las localizaciones de los fotogramas claves sobre un eje temporal y datos de valores de claves que indican la rotación del objeto en las localizaciones temporales indicadas por los datos de claves a lo largo de una primera trayectoria de animación, comprendiendo el aparato:

un extractor de puntos de ruptura (42, 46, 48) que está dispuesto para extraer, a partir de la entrada de orientación dentro del extractor de puntos de ruptura que definen el movimiento del objeto en 3D a lo largo de una primera trayectoria de animación, los datos de claves y los datos de valores de claves en un número mínimo de puntos de ruptura a lo largo del eje temporal, que pueden lograr un error no mayor que un límite de error predeterminado entre los datos de valores de claves a lo largo de la primera trayectoria de animación y los datos de valores de claves a lo largo de una segunda trayectoria de animación, en el que la segunda trayectoria de animación es la trayectoria de animación definida por los datos de claves extraídos y los datos de valores de claves en los puntos de ruptura extraídos.
un codificador de datos de valores de claves (200) que está dispuesto para codificar la entrada de datos de claves extraídos a partir del extractor de puntos de ruptura; y
un codificador de datos de valores de claves (300) que está dispuesto para codificar la entrada de datos de valores de claves extraídos a partir del extractor de puntos de ruptura generando los datos diferenciales rotacionales;
en el que el codificador de valores de claves (300) comprende:
un generador de datos diferenciales rotacionales (1300) que genera, usando un valor de la transformación rotacional de un fotograma clave actual y un valor de la transformación rotacional restaurada de un fotograma clave anterior, un valor diferencial rotacional usado para girar el objeto tanto como la diferencia entre la transformación rotacional aplicada al objeto en el fotograma clave actual por los datos de valores de claves y la transformación rotacional aplicada al objeto en el fotograma clave anterior por los datos de valores de claves, y saca los datos diferenciales rotacionales por la cuantización de los valores diferenciales rotacionales; y
un codificador de entropía (1450) que codifica en entropía los datos diferenciales rotacionales.

2. El aparato de la reivindicación 1 que comprende además

un dispositivo de re-muestreo (43) que muestrea la primera trayectoria de animación dentro de un número predeterminado de secciones que tienen un intervalo de una magnitud predeterminada de tiempo y saca un interpolador de orientación incluyendo datos de las claves re-muestreadas y datos de valores de claves re-muestreadas; y
un selector (41) que saca un interpolador de orientación introducido en el mismo al dispositivo de re-muestreo o al extractor de puntos de ruptura dependiendo de una señal de entrada externa.

3. El aparato de la reivindicación 1 que comprende además un dispositivo de re-muestreo (45) que muestrea la primera trayectoria de animación dentro de un número predeterminado de secciones que tienen un intervalo de una cantidad de tiempo predeterminada y saca un interpolador de orientación incluyendo los datos de claves re-muestreadas y los datos de valores de claves re-muestreados,
en el que el extractor de puntos de ruptura (46) extrae los puntos de ruptura desde una trayectoria de animación constituida por un interpolador de orientación introducido desde el dispositivo de re-muestreo.

4. El aparato de la reivindicación 1 que comprende además un dispositivo de re-muestreo (49) que muestrea una trayectoria de animación constituida por un interpolador de orientación extraída del extractor de puntos de ruptura (48) en un número predeterminado de secciones que tienen un intervalo de una cantidad de tiempo predeterminada y saca un interpolador de orientación incluyendo datos de claves re-muestreados y datos de valores de claves re-muestreados al codificador de datos de claves y al codificador de datos de valores de claves.

5. El aparato de cualquiera de las reivindicaciones 2 a 4, en el que el dispositivo de re-muestreo (43, 45, 49) divide una trayectoria de animación constituida por datos de claves y datos de valores de claves de un interpolador de orientación en un número predeterminado de secciones que tienen un intervalo de una cantidad predeterminada de tiempo, saca los puntos del final de cada una de las secciones como datos de claves a codificar y saca los datos de valores de claves existentes sobre la trayectoria de animación en cada una de las secciones como datos de valores de claves a codificar.

6. El aparato de cualquiera de las reivindicaciones 1 a 4, en el que el extractor de puntos de ruptura (42, 46, 48) comprende:

un interpolador lineal (42a, 46a, 48a) que extrae un punto de comienzo de la trayectoria y un punto de final de la trayectoria de una trayectoria de animación de entrada, selecciona puntos de la trayectoria entre los puntos de comienzo y de final de la trayectoria, e interpola de forma lineal esférica otros puntos de la trayectoria que no se han seleccionado aún, usando los puntos de la trayectoria seleccionados y los puntos de la trayectoria extraídos;

un calculador de errores (42b, 46b, 48b) que calcula un error entre la trayectoria de animación de entrada y la trayectoria de animación interpolada generada por el interpolador lineal usando la interpolación; y
una unidad de determinación (42c, 46c, 48c) que extrae puntos de ruptura, por los cuales un error entre la trayectoria de animación de entrada y la trayectoria de animación interpolada puede minimizarse, y saca los puntos de ruptura extraídos si los errores correspondientes no son mayores de un límite de error predeterminado.

7. El aparato de la reivindicación 6, en el que si el error entre la trayectoria de animación de entrada y la trayectoria de animación interpolada constituida por los puntos de ruptura extraídos, es mayor de un límite de error predeterminado, el interpolador lineal selecciona todos los puntos de la trayectoria excepto los puntos de ruptura introducidos desde la unidad de determinación, uno por uno y realiza la interpolación lineal esférica sobre los puntos de la trayectoria seleccionados.

8. El aparato de la reivindicación 6, en el que el calculador de errores (42b, 46b, 48b) divide la trayectoria de animación de entrada y la trayectoria de animación generada por la interpolación lineal esférica dentro de un número de secciones predeterminado en base a una componente de referencia que constituye sus puntos de trayectoria y calcula un error entre la trayectoria de animación de entrada y la trayectoria de animación generada en cada una de las secciones por la medición de un área de cada una de las secciones.

9. El aparato de la reivindicación 1, en el que el codificador de datos de claves (200) cuantiza la entrada de datos de claves desde el extractor de puntos de ruptura usando bits de cuantización predeterminados, genera datos diferenciales realizando una operación de DPCM predeterminada sobre los datos de claves cuantizados, y codifica los datos diferenciales.

10. El aparato de la reivindicación 1, en el que el codificador de datos de claves (200) comprende:

un primer dispositivo de cuantización que cuantiza los datos de claves de un interpolador de orientación usando bits de cuantización predeterminados;
un primer procesador de DPCM que genera datos diferenciales de los datos de claves cuantizados;
un procesador de DND que realiza una operación de DND sobre los datos diferenciales dependiendo de la relación entre los datos diferenciales y el valor máximo y el valor mínimo de entre ellos; y
un primer codificador de entropía que codifica en entropía la entrada de datos diferenciales procedente del procesador de DND;
en el que la operación de DND es una operación de división seguida de una operación de división hacia arriba o una operación de división hacia abajo, estando dada la operación de división por

$$\textit{división}\ (nQKey_j,\ nKeyMax)$$

$$= nQKey_j - (nKeyMax + 1) \qquad (si\ nQKey_j > nKeyMax / 2)$$

$$= nQkey_j \qquad (si\ nQKey_j \leq nKeyMax / 2)$$

11. El aparato de la reivindicación 1, en el que el codificador de datos de claves comprende además un codificador de claves lineal, que identifica y decodifica una región donde los datos de claves aumentan linealmente de entre todos los datos de claves introducidos en el mismo.

12. El aparato de la reivindicación 10, en el que el codificador de datos de claves (200) comprende además:

un dispositivo de desplazamiento que obtiene un dato diferencial (modo) que tiene la frecuencia más alta de entre los datos diferenciales introducidos desde el primer procesador de DPCM y resta el modo de los datos diferenciales; y
un procesador de plegado que convierte los datos diferenciales desplazados en números positivos o números negativos, y un procesador de DND que selecciona una de las entradas de datos diferenciales desde el dispositivo de desplazamiento, la entrada de datos diferenciales desde el procesador de plegado, y los datos diferenciales con la DND dependiendo del número de bits requeridos para la codificación y saca los datos diferenciales seleccionados.

13. El aparato de la reivindicación 1, en el que el generador de datos diferenciales rotacionales comprende:

un dispositivo de cuantización que genera datos diferenciales rotacionales por cuantización de los tres valores componentes del valor diferencial rotacional;
un dispositivo de ajuste de los datos cuantizados que ajusta los tres valores componentes de los datos diferenciales rotacionales que entran en el mismo;
un dispositivo de cuantización inversa que realiza la cuantización inversa de los valores componentes ajustados;
un restaurador de valores diferenciales rotacionales que restaura un valor componente, que no se ha cuantizado, usando los tres valores componentes con cuantización inversa y de este modo genera un valor diferencial rotacional restaurado; y

una unidad de mediciones de error que mide un error entre la entrada de valores diferenciales rotacionales dentro del dispositivo de cuantización y el valor diferencial rotacional restaurado y saca los datos diferenciales rotacionales que tienen los valores componentes ajustados de modo que puede minimizarse el error.

14. El aparato de la reivindicación 1, en el que el generador de datos diferenciales rotacionales comprende;

un primer multiplicador de cuaternios (1310) que genera los valores diferenciales rotacionales usando el valor de la transformación rotacional del fotograma clave actual y el valor de la transformación rotacional restaurado del fotograma clave anterior;
un dispositivo de cuantización (1340) que genera datos diferenciales rotacionales por la cuantización del valor diferencial rotacional;
un dispositivo de cuantización inversa (1350) que genera un valor diferencial rotacional restaurado por la cuantización inversa de los datos diferenciales rotacionales; y
un segundo multiplicador de cuaternios (1370) que genera un valor diferencial rotacional restaurado del fotograma clave actual por la multiplicación de cuaternios del valor diferencial rotacional restaurado por un valor de la transformación rotacional del fotograma clave anterior.

15. El aparato de la reivindicación 1, en el que el generador de datos diferenciales rotacionales comprende:

un detector de errores de la dirección de rotación (1830) que detecta si ha ocurrido o no un error de la dirección de rotación de modo que la dirección de rotación original del objeto es opuesta a la dirección de rotación decodificada del objeto, en base al valor de la transformación rotacional del fotograma clave actual y el valor de la transformación rotacional restaurada del fotograma clave anterior;
un corrector de la dirección de rotación (1815) que ajusta el valor diferencial rotacional, de modo que la dirección de rotación decodificada del objeto puede ser la misma que la dirección de rotación original del objeto; y
un selector de la dirección de rotación (1835) que selecciona bien el valor diferencial de rotación o el valor diferencial de rotación introducido desde el corrector de la dirección de rotación como datos diferenciales a cuantizar, dependiendo del resultado de la detección introducida desde el detector de errores de la dirección de rotación.

16. Un procedimiento para la decodificación de un flujo de bits, en el cual se codifica un interpolador de orientación usado en la animación de un fotograma clave de un objeto de 3D, incluyendo el interpolador de orientación datos de claves que indican las localizaciones de los fotogramas claves sobre un eje temporal e indicando los datos de valores de claves la rotación de un objeto, comprendiendo el procedimiento:

generar datos diferenciales por la decodificación de entropía del flujo de bits;
generar datos de claves cuantizados realizando las operaciones de la DND inversa, el plegado inverso, el desplazamiento inverso, y la DPCM inversa sobre los datos diferenciales si el orden de DND (nDNDOrder) de los datos diferenciales no es menor de 1, realizando las operaciones del plegado inverso, el desplazamiento inverso, y la DPCM inversa sobre los datos diferenciales si el orden de DND de los datos diferenciales es 0, y realizando las operaciones de desplazamiento inverso y DPCM inversa sobre los datos diferenciales si el orden de DND de los datos diferenciales es -1;
reconstruir los datos de claves reconstruidos por la cuantización inversa de los datos de claves cuantizados;
decodificar en entropía los datos de valores de claves a partir del flujo de bits, para generar datos diferenciales rotacionales cuantizados por la realización de la operación de DPCM circular inversa sobre los datos de valores de claves si el orden de la DPCM (nKVDPCMOrder) incluido en una cabecera del valor de clave es 1, para generar los datos diferenciales rotacionales cuantizados si el orden de la DPCM es 0;
generar un valor diferencial rotacional de un fotograma clave actual usado para girar un objeto tanto como la diferencia entre las transformaciones rotacionales aplicadas al objeto por los datos de valores de claves de cuaternios de cada uno de los fotogramas claves por la cuantización inversa de los datos diferenciales rotacionales cuantizados;
generar un valor de la transformación rotacional del fotograma clave actual por la multiplicación de cuaternios del valor diferencial rotacional del fotograma clave actual por un valor de la transformación rotacional decodificada de un fotograma clave anterior;
generar un interpolador de orientación sintetizando los datos de claves decodificados y los datos de valores de claves que se interpolan de forma lineal esférica usando los datos de claves decodificados y los datos de valores de claves decodificados, en el que los datos de valores de claves de la trama actual se restauran interpolando el fotograma clave anterior y el fotograma clave posterior si un indicador de la selección de clave del fotograma clave actual se fija a un valor de 0; y sacar el interpolador de orientación.

17. Un programa de ordenador que comprende un medio de código de programa de ordenador para la realización de todas las etapas de la reivindicación 16 cuando dicho programa corre sobre un ordenador.

18. Un programa de ordenador de acuerdo con la reivindicación 17 realizado sobre un medio legible por ordenador.

FIG. 1

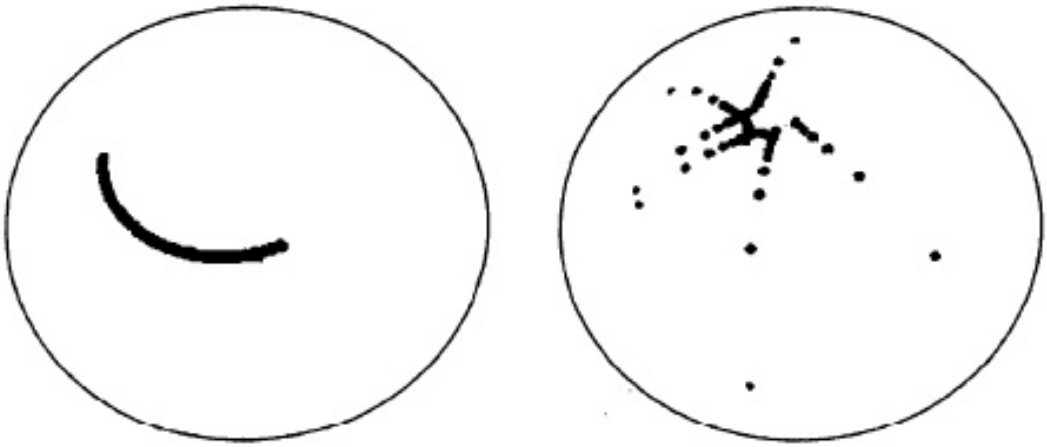

FIG. 2A

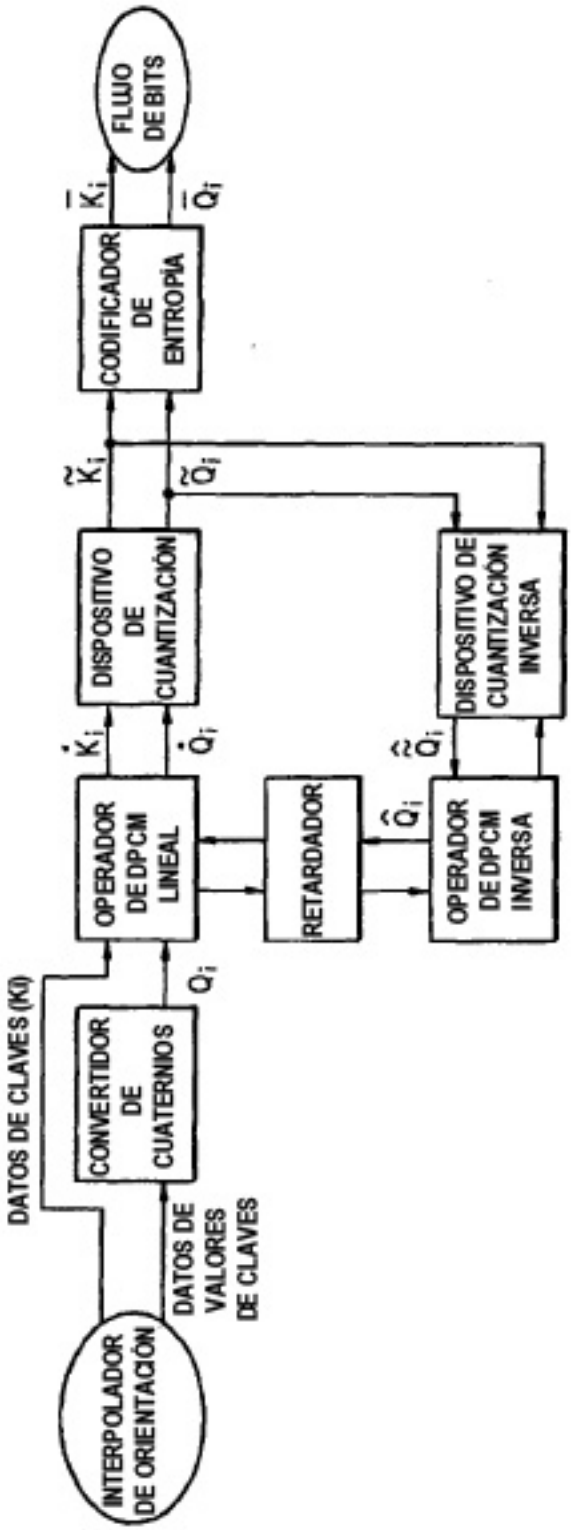
FLUJO DE BITS
CODIFICADOR DE ENTROPÍA
DISPOSITIVO DE CUANTIZACIÓN
DISPOSITIVO DE CUANTIZACIÓN INVERSA
OPERADOR DE DPCM LINEAL
RETARDADOR
OPERADOR DE DPCM INVERSA
CONVERTIDOR DE CUATERNIOS
INTERPOLADOR DE ORIENTACIÓN
DATOS DE CLAVES (Ki)
DATOS DE VALORES DE CLAVES

FIG. 2B

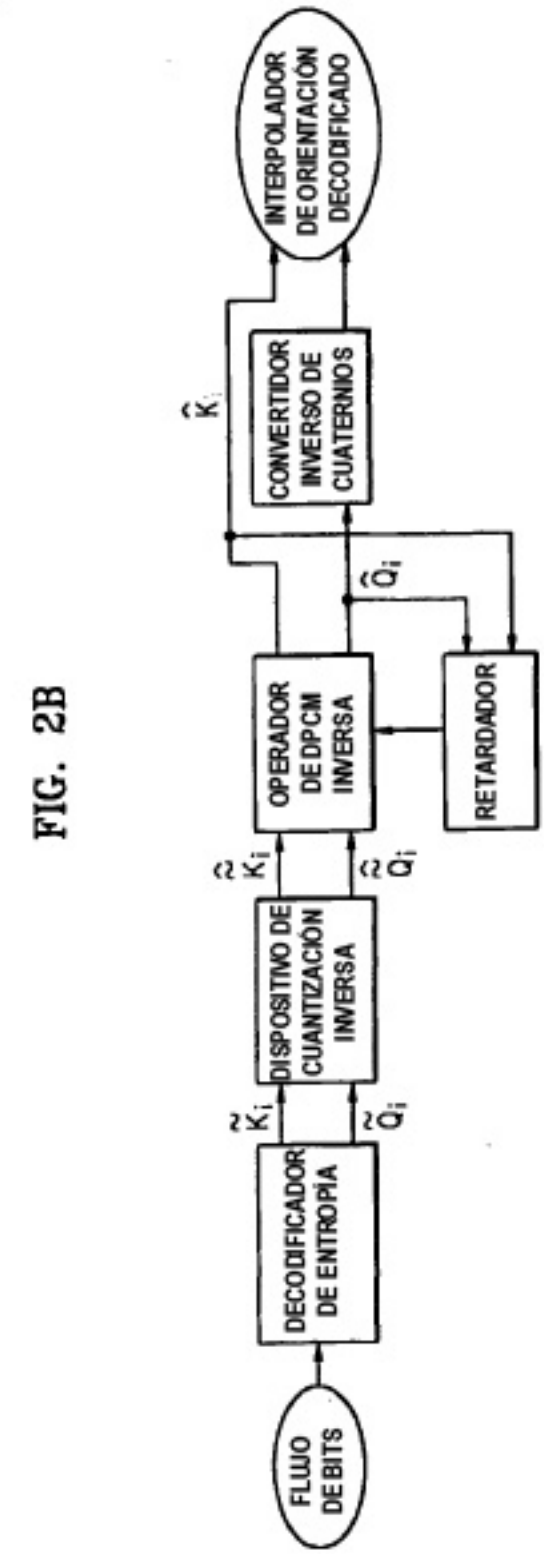
FLUJO DE BITS
DECODIFICADOR DE ENTROPÍA
DISPOSITIVO DE CUANTIZACIÓN INVERSA
OPERADOR DE DPCM INVERSA
RETARDADOR
CONVERTIDOR INVERSO DE CUATERNIOS
INTERPOLADOR DE ORIENTACIÓN DECODIFICADO

FIG. 3A

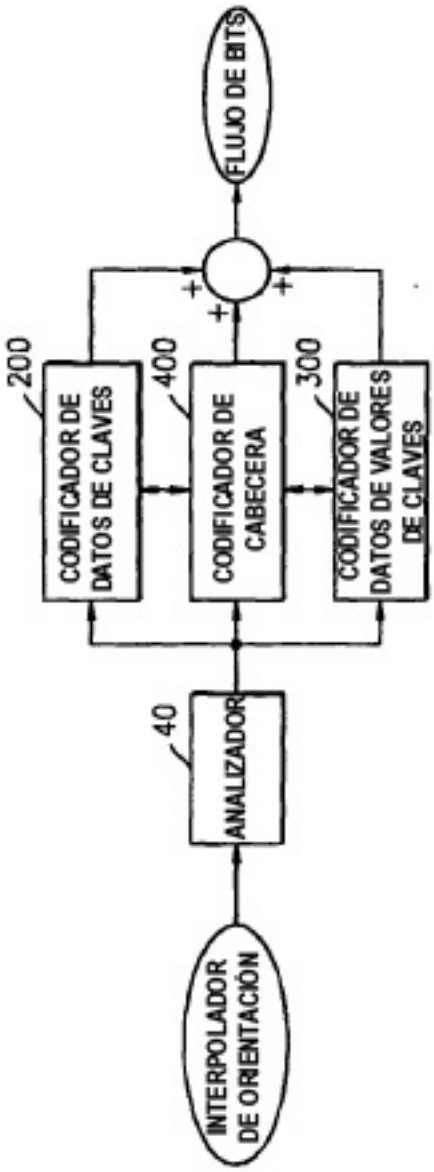
FLUJO DE BITS
+
+
+
200
CODIFICADOR DE DATOS DE CLAVES
400
CODIFICADOR DE CABECERA
300
CODIFICADOR DE DATOS DE VALORES DE CLAVES
40
ANALIZADOR
INTERPOLADOR DE ORIENTACIÓN

FIG. 3B

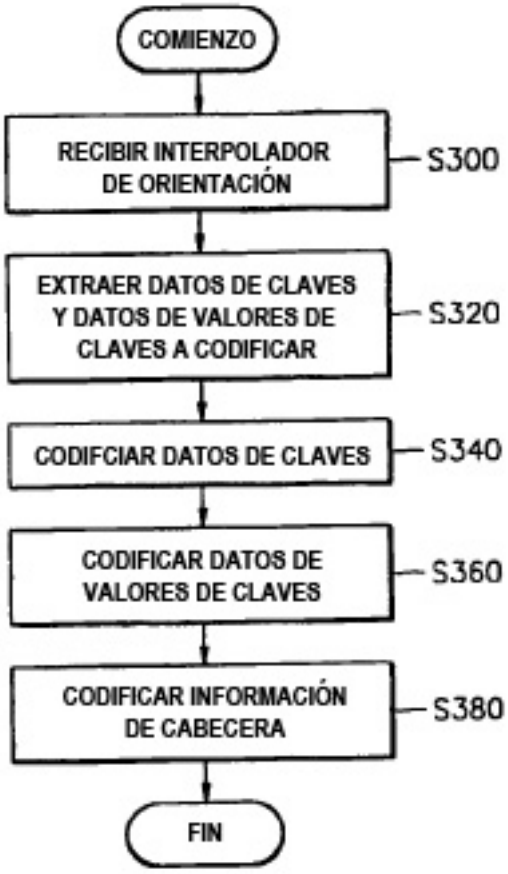
COMIENZO
RECIBIR INTERPOLADOR DE ORIENTACIÓN
S300
EXTRAER DATOS DE CLAVES Y DATOS DE VALORES DE CLAVES A CODIFICAR
S320
CODIFCIAR DATOS DE CLAVES
S340
CODIFICAR DATOS DE VALORES DE CLAVES
S360
CODIFICAR INFORMACIÓN DE CABECERA
S380
FIN

FIG. 4A

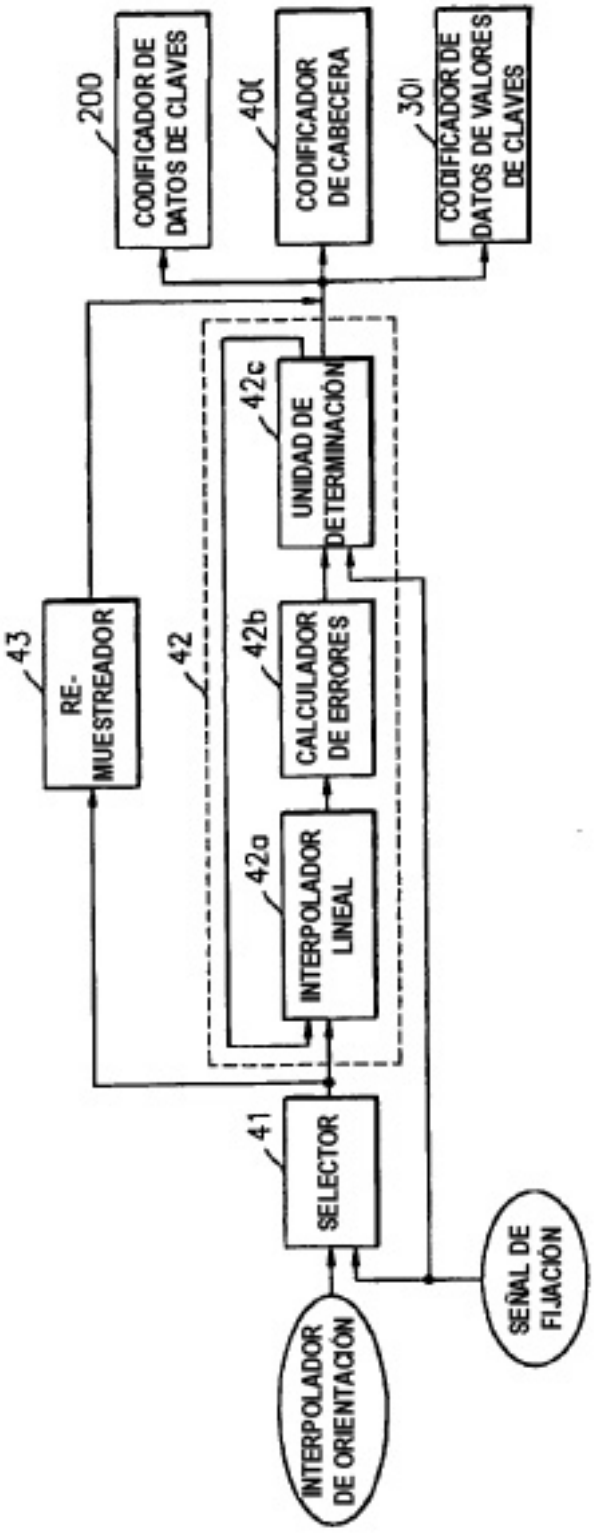
200
CODIFICADOR DE DATOS DE CLAVES
40(
CODIFICADOR DE CABECERA
30(
CODIFICADOR DE DATOS DE VALORES DE CLAVES
42c
UNIDAD DE DETERMINACIÓN
42b
CALCULADOR DE ERRORES
42a
INTERPOLADOR LINEAL
42
43
RE-MUESTREADOR
41
SELECTOR
INTERPOLADOR DE ORIENTACIÓN
SEÑAL DE FIJACIÓN

FIG. 4B

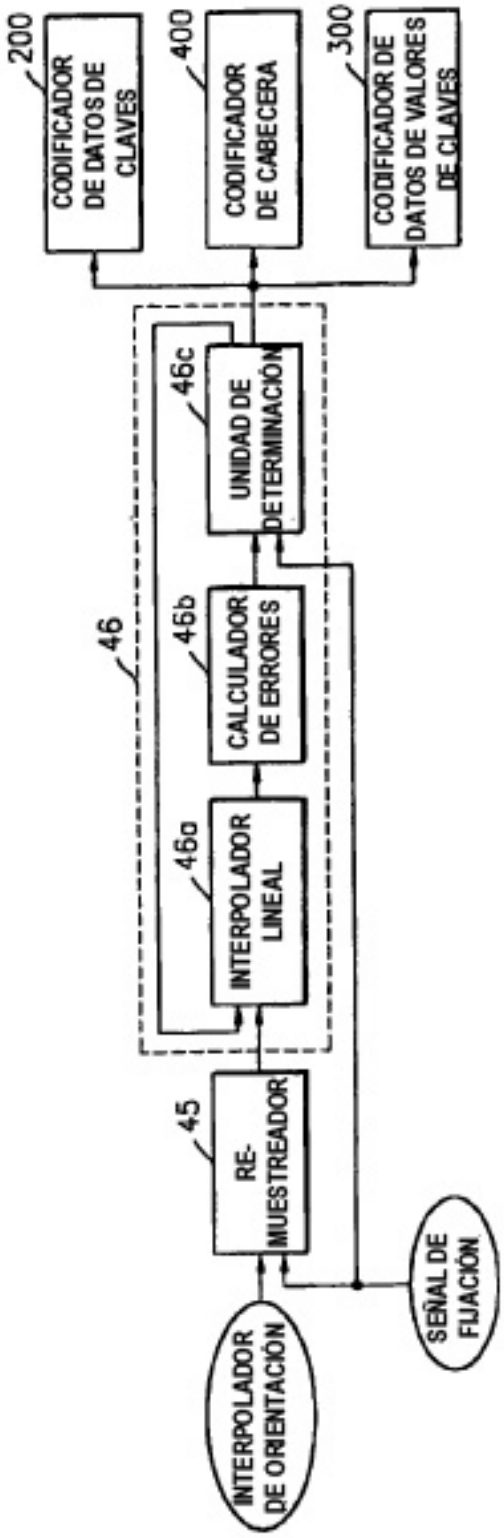
200
CODIFICADOR DE DATOS DE CLAVES
400
CODIFICADOR DE CABECERA
300
CODIFICADOR DE DATOS DE VALORES DE CLAVES
46
46c
UNIDAD DE DETERMINACIÓN
46b
CALCULADOR DE ERRORES
46a
INTERPOLADOR LINEAL
45
RE-MUESTREADOR
INTERPOLADOR DE ORIENTACIÓN
SEÑAL DE FIJACIÓN

FIG. 4C

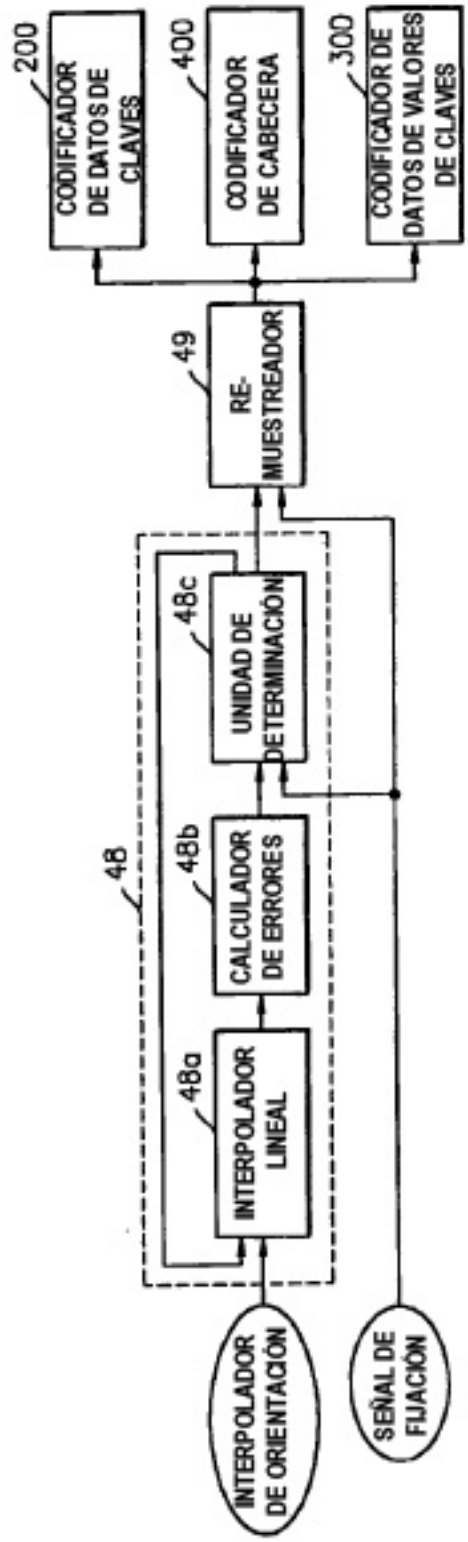
200
CODIFICADOR DE DATOS DE CLAVES
400
CODIFICADOR DE CABECERA
300
CODIFICADOR DE DATOS DE VALORES DE CLAVES
49
RE-MUESTREADOR
48
48a
INTERPOLADOR LINEAL
48b
CALCULADOR DE ERRORES
48c
UNIDAD DE DETERMINACIÓN
INTERPOLADOR DE ORIENTACIÓN
SEÑAL DE FIJACIÓN

FIG. 5A

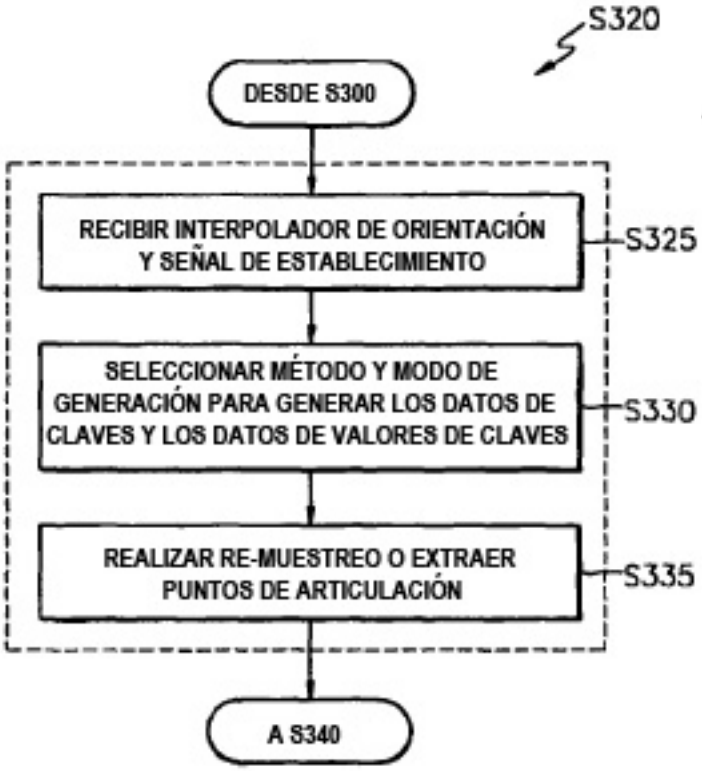
S320
DESDE S300
RECIBIR INTERPOLADOR DE ORIENTACIÓN Y SEÑAL DE ESTABLECIMIENTO
S325
SELECCIONAR MÉTODO Y MODO DE GENERACIÓN PARA GENERAR LOS DATOS DE CLAVES Y LOS DATOS DE VALORES DE CLAVES
S330
REALIZAR RE-MUESTREO O EXTRAER PUNTOS DE ARTICULACIÓN
S335
A S340

FIG. 5B

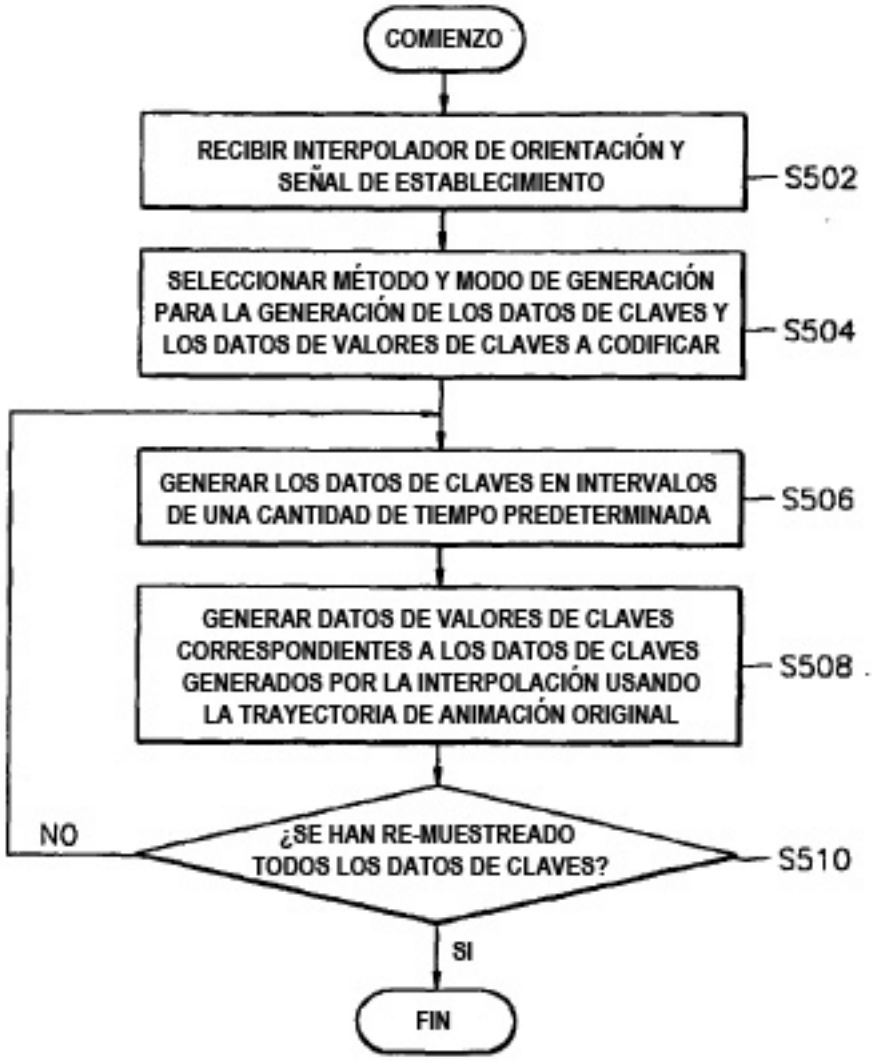
COMIENZO
RECIBIR INTERPOLADOR DE ORIENTACIÓN Y SEÑAL DE ESTABLECIMIENTO
S502
SELECCIONAR MÉTODO Y MODO DE GENERACIÓN PARA LA GENERACIÓN DE LOS DATOS DE CLAVES Y LOS DATOS DE VALORES DE CLAVES A CODIFICAR
S504
GENERAR LOS DATOS DE CLAVES EN INTERVALOS DE UNA CANTIDAD DE TIEMPO PREDETERMINADA
S506
GENERAR DATOS DE VALORES DE CLAVES CORRESPONDIENTES A LOS DATOS DE CLAVES GENERADOS POR LA INTERPOLACIÓN USANDO LA TRAYECTORIA DE ANIMACIÓN ORIGINAL
S508
¿SE HAN RE-MUESTREADO TODOS LOS DATOS DE CLAVES?
S510
NO
SI
FIN

FIG. 5C


COMIENZO
RECIBIR INTERPOLADOR DE ORIENTACIÓN Y ERROR CRÍTICO (e_{th})
S520
EXTRAER LOS PUNTOS PRIMERO Y FINAL DE LA TRAYECTORIA ORIGINAL
S522
SELECCIONAR LOS PUNTOS NO EXTRAÍDOS DE LA TRAYECTORIA UNO POR UNO
S524
INERPOLAR LOS PUNTOS DE LA TRAYECTORIA NO SELECCIONADOS USANDO LOS PUNTOS SELECCIONADOS DE LA TRAYECTORIA
S526
CALCULAR EL ERROR ENTRE LA TRAYECTORIA ORIGINAL Y LA TRAYECTORIA COMPRENDIDA DE LOS PUNTOS DE LA TRAYECTORIA SELECCIONADOS
S528
¿SE HA REALIZADO EL CÁLCULO DE ERROR SOBRE TODOS LOS PUNTOS NO EXTRAÍDOS DE LA TRAYECTORIA?
S530
NO
SI
EXTRAER LOS PUNTOS DE LA TRAYECTORIA QUE CONDUCEN A UN ERROR MÍNIMO, SELECCIONAR EL ERROR (e) Y AUMENTAR EL VALOR DEL CONTADOR EN 1
S532
SI
(i<n) Y (e>e_{th})
S534
NO
FIN

FIG. 6A

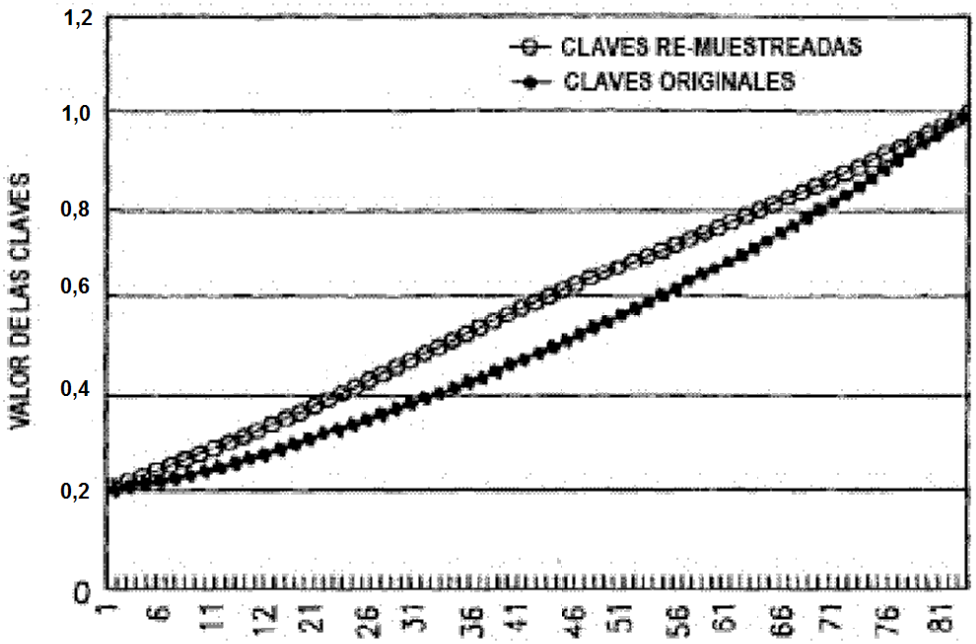
CLAVES RE-MUESTREADAS
CLAVES ORIGINALES
VALOR DE LAS CLAVES
1,2
1,0
0,8
0,6
0,4
0,2
0
1
6
11
12
21
26
31
36
41
46
51
56
61
66
71
76
81

FIG. 6B

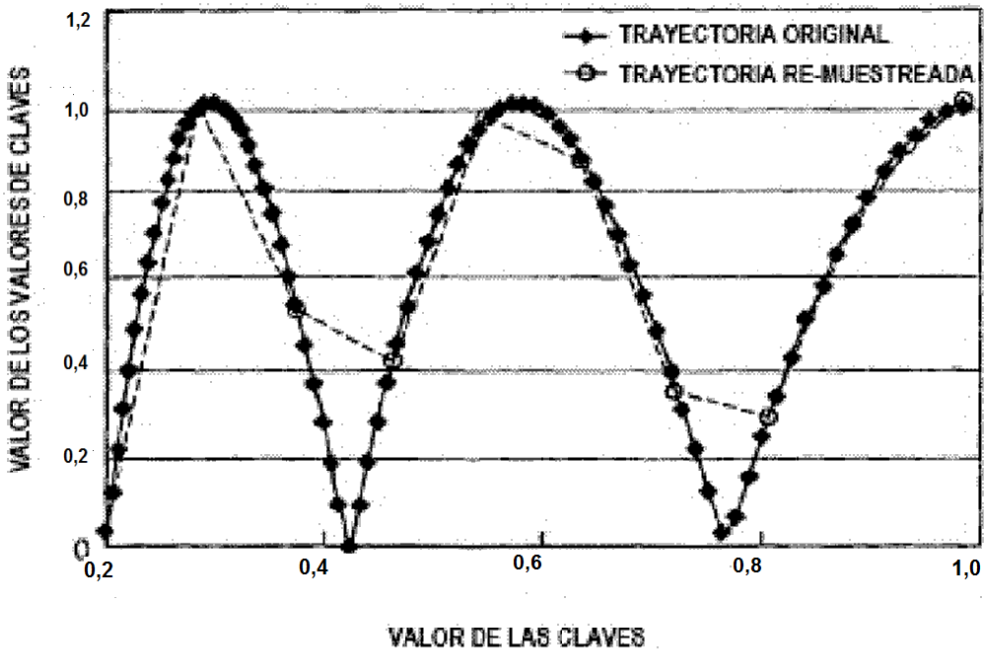
TRAYECTORIA ORIGINAL
TRAYECTORIA RE-MUESTREADA
VALOR DE LOS VALORES DE CLAVES
1,2
1,0
0,8
0,6
0,4
0,2
0
0,2
0,4
0,6
0,8
1,0
VALOR DE LAS CLAVES

FIG. 7A

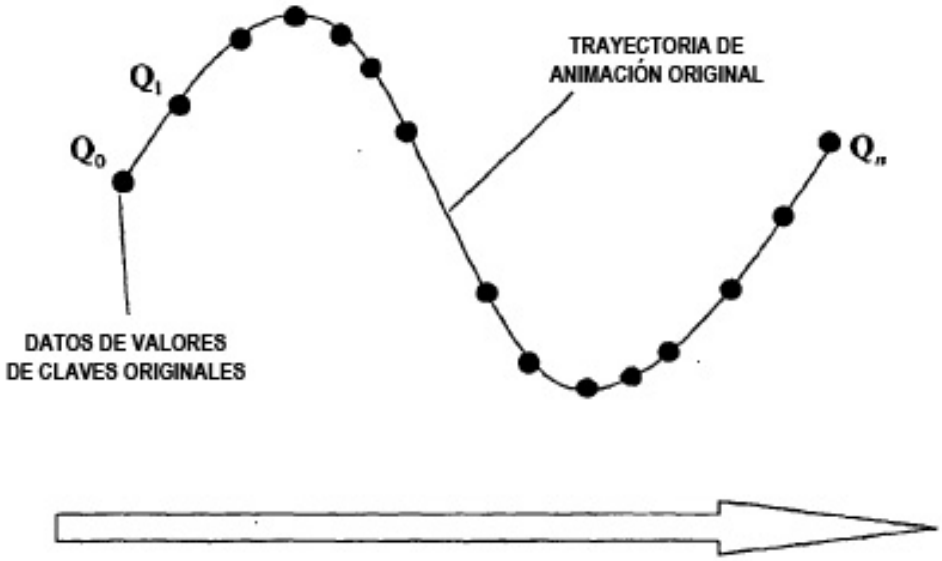
TRAYECTORIA DE
ANIMACIÓN ORIGINAL
Q_1
Q_0
Q_n
DATOS DE VALORES
DE CLAVES ORIGINALES

FIG. 7B

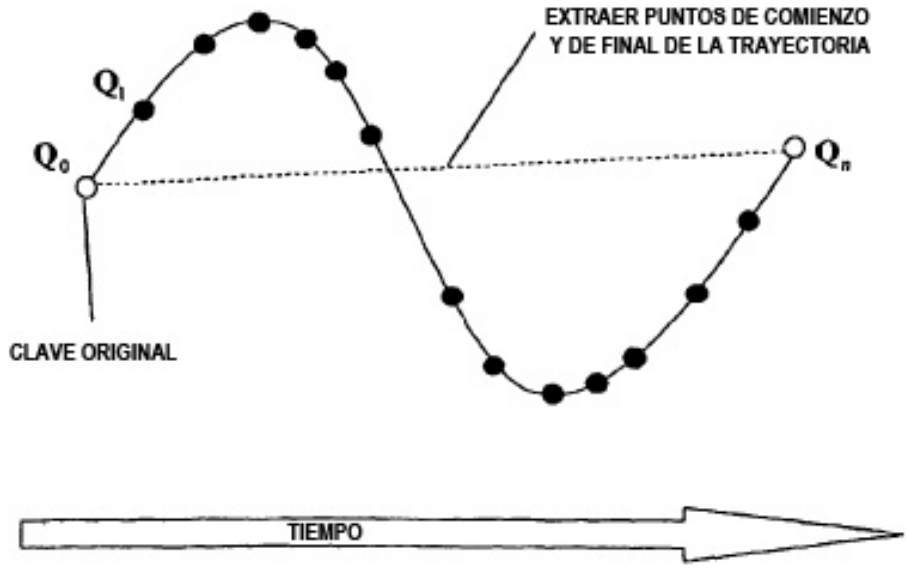
EXTRAER PUNTOS DE COMIENZO
Y DE FINAL DE LA TRAYECTORIA
Q_1
Q_0
Q_n
CLAVE ORIGINAL
TIEMPO

FIG. 7C

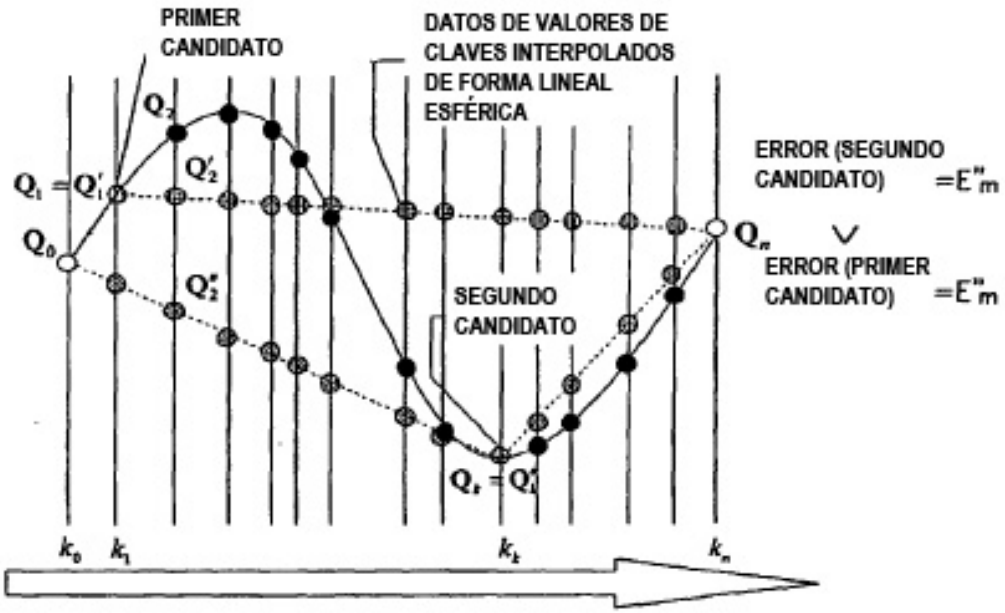
PRIMER CANDIDATO
DATOS DE VALORES DE CLAVES INTERPOLADOS DE FORMA LINEAL ESFÉRICA
ERROR (SEGUNDO CANDIDATO) =E"m
ERROR (PRIMER CANDIDATO) =E"m
SEGUNDO CANDIDATO
Q0
Q1 = Q'1
Q2
Q'2
Q"2
Qn
Qk = Q"1
k0
k1
kk
kn

FIG. 7D

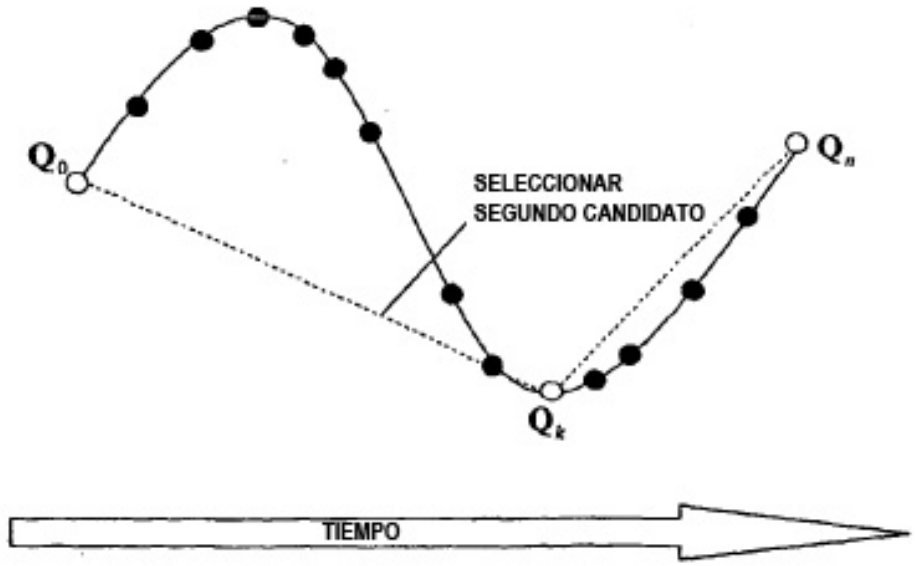
Q0
Qn
SELECCIONAR SEGUNDO CANDIDATO
Qk
TIEMPO

FIG. 7E

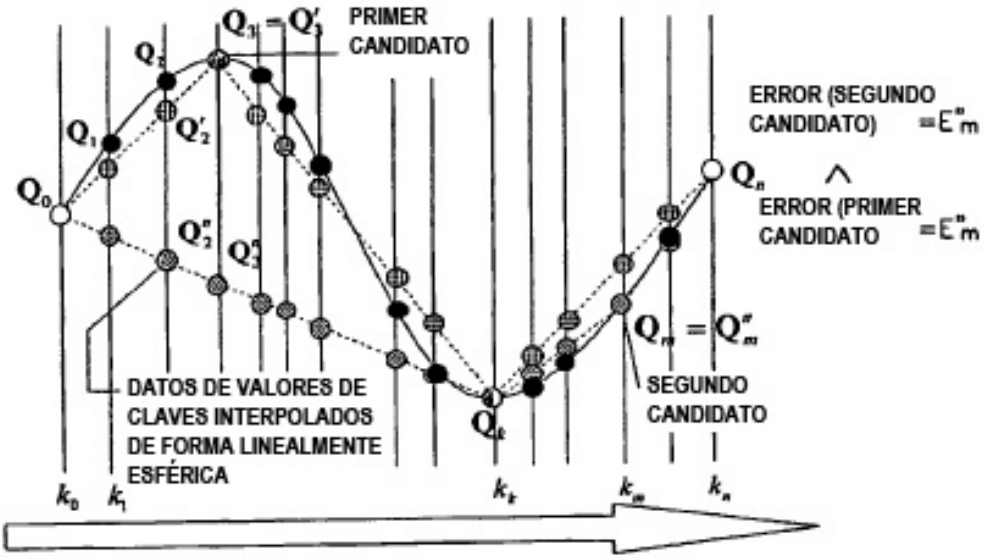
PRIMER CANDIDATO
ERROR (SEGUNDO CANDIDATO) $=E''_m$
ERROR (PRIMER CANDIDATO $=E'_m$
DATOS DE VALORES DE CLAVES INTERPOLADOS DE FORMA LINEALMENTE ESFÉRICA
SEGUNDO CANDIDATO

FIG. 7F

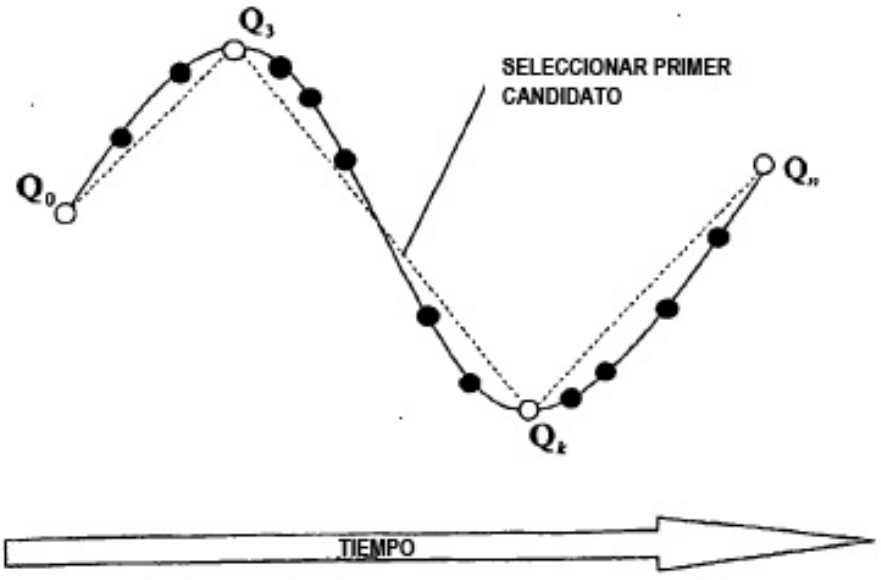
SELECCIONAR PRIMER CANDIDATO
TIEMPO

FIG. 8

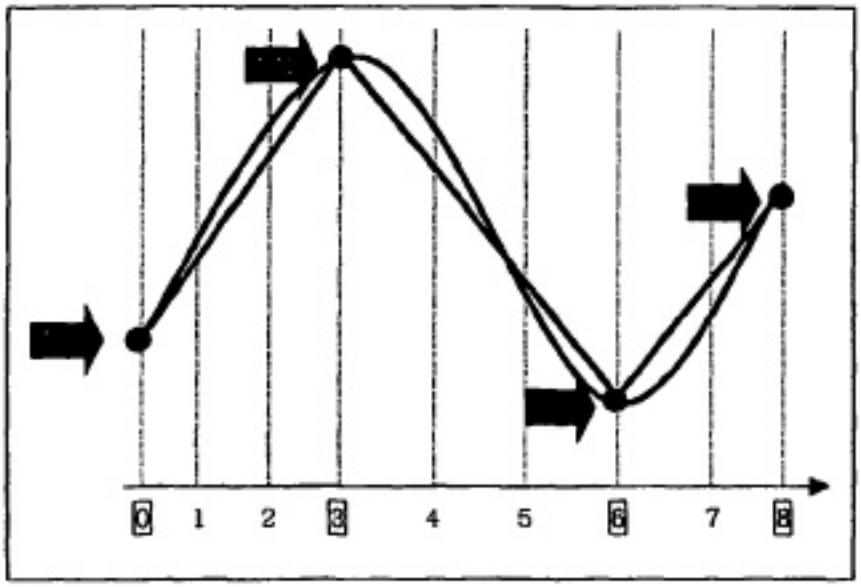

FIG. 9A

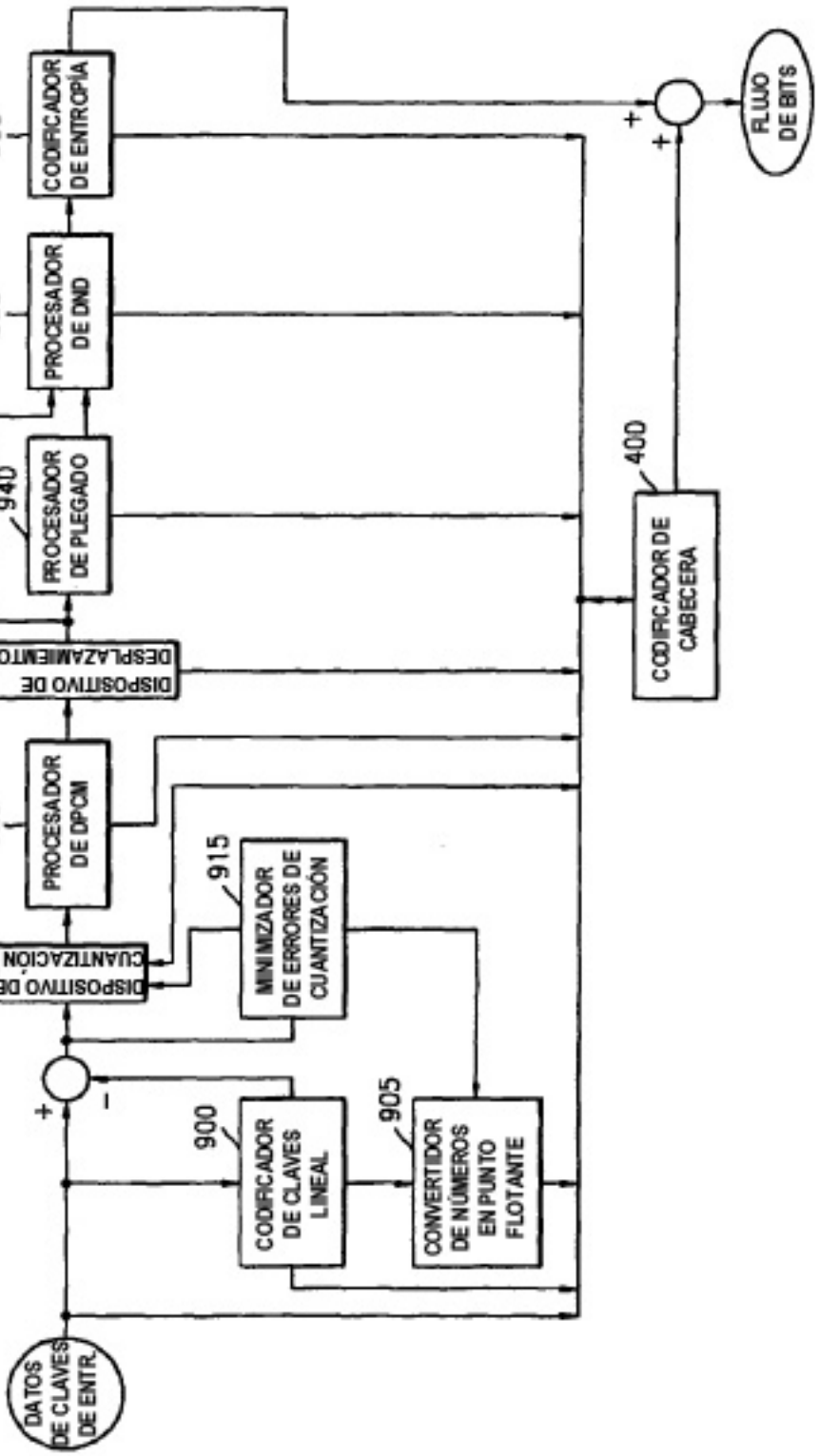
DATOS DE CLAVES DE ENTR.
CODIFICADOR DE CLAVES LINEAL
900
CONVERTIDOR DE NÚMEROS EN PUNTO FLOTANTE
905
MINIMIZADOR DE ERRORES DE CUANTIZACIÓN
915
DISPOSITIVO DE CUANTIZACIÓN
910
PROCESADOR DE DPCM
920
DISPOSITIVO DE DESPLAZAMIENTO
930
PROCESADOR DE PLEGADO
940
PROCESADOR DE DND
950
CODIFICADOR DE ENTROPÍA
960
CODIFICADOR DE CABECERA
400
FLUJO DE BITS

FIG. 9B

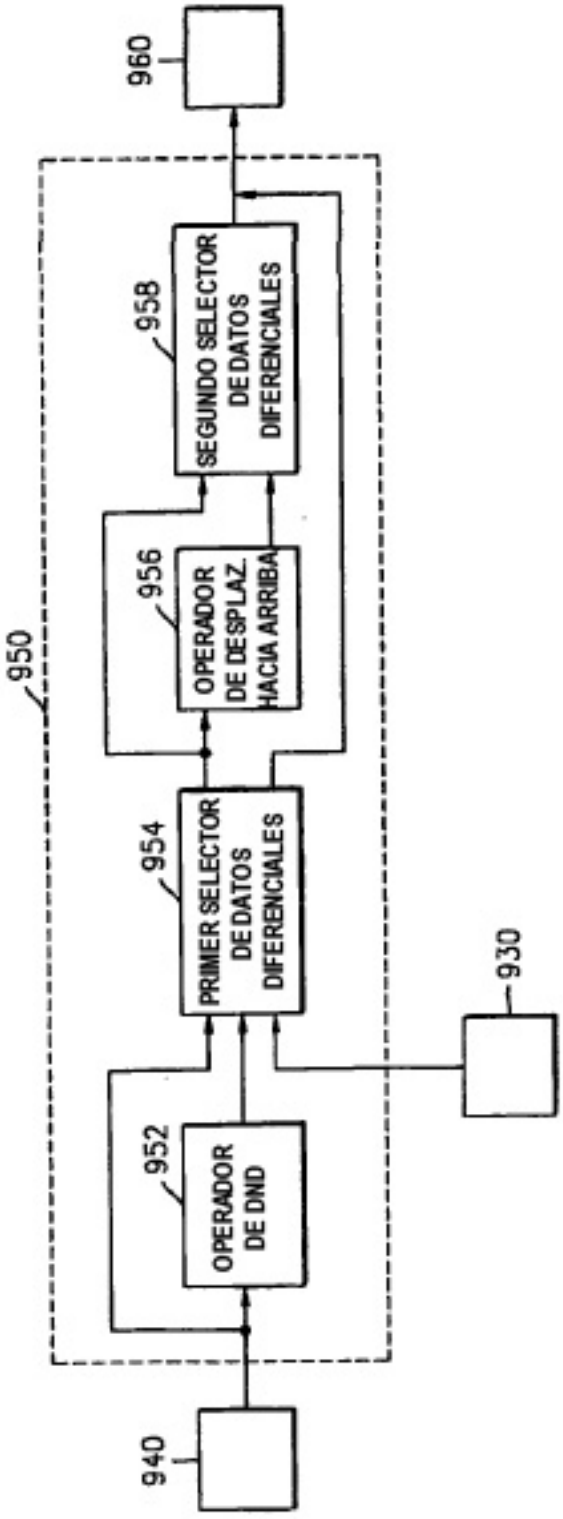
960
950
958
SEGUNDO SELECTOR DE DATOS DIFERENCIALES
956
OPERADOR DE DESPLAZ. HACIA ARRIBA
954
PRIMER SELECTOR DE DATOS DIFERENCIALES
930
952
OPERADOR DE DND
940

FIG. 10A

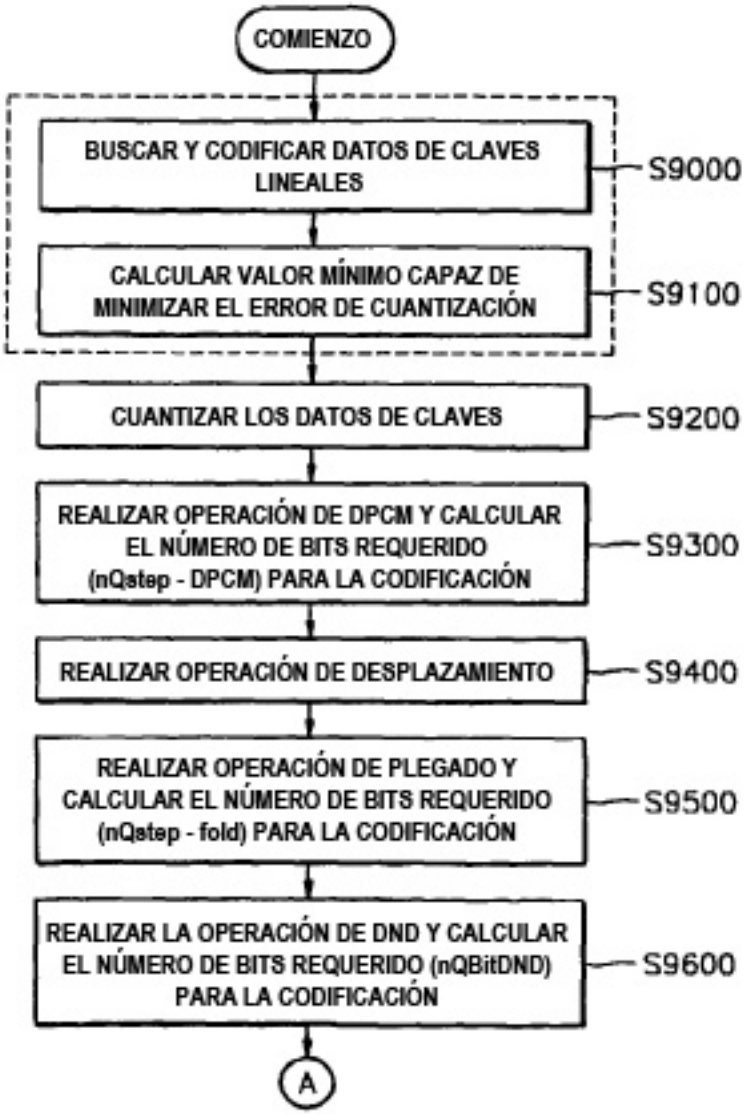
COMIENZO
BUSCAR Y CODIFICAR DATOS DE CLAVES LINEALES
S9000
CALCULAR VALOR MÍNIMO CAPAZ DE MINIMIZAR EL ERROR DE CUANTIZACIÓN
S9100
CUANTIZAR LOS DATOS DE CLAVES
S9200
REALIZAR OPERACIÓN DE DPCM Y CALCULAR EL NÚMERO DE BITS REQUERIDO (nQstep - DPCM) PARA LA CODIFICACIÓN
S9300
REALIZAR OPERACIÓN DE DESPLAZAMIENTO
S9400
REALIZAR OPERACIÓN DE PLEGADO Y CALCULAR EL NÚMERO DE BITS REQUERIDO (nQstep - fold) PARA LA CODIFICACIÓN
S9500
REALIZAR LA OPERACIÓN DE DND Y CALCULAR EL NÚMERO DE BITS REQUERIDO (nQBitDND) PARA LA CODIFICACIÓN
S9600
A

FIG. 10B

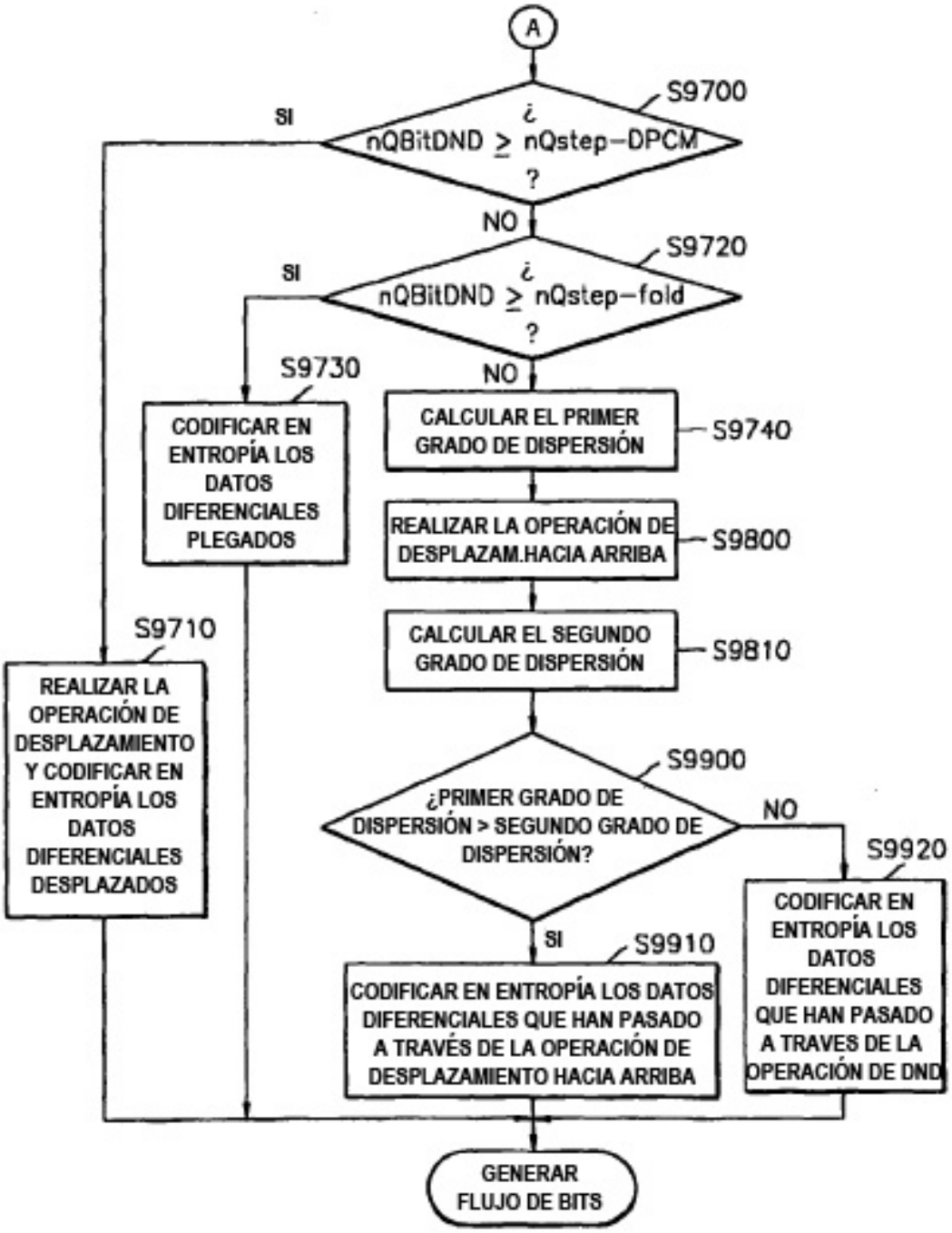
A
S9700
¿ nQBitDND ≥ nQstep-DPCM ?
SI
NO
S9720
¿ nQBitDND ≥ nQstep-fold ?
SI
NO
S9730
CODIFICAR EN ENTROPÍA LOS DATOS DIFERENCIALES PLEGADOS
S9740
CALCULAR EL PRIMER GRADO DE DISPERSIÓN
S9800
REALIZAR LA OPERACIÓN DE DESPLAZAM.HACIA ARRIBA
S9810
CALCULAR EL SEGUNDO GRADO DE DISPERSIÓN
S9710
REALIZAR LA OPERACIÓN DE DESPLAZAMIENTO Y CODIFICAR EN ENTROPÍA LOS DATOS DIFERENCIALES DESPLAZADOS
S9900
¿PRIMER GRADO DE DISPERSIÓN > SEGUNDO GRADO DE DISPERSIÓN?
NO
S9920
CODIFICAR EN ENTROPÍA LOS DATOS DIFERENCIALES QUE HAN PASADO A TRAVES DE LA OPERACIÓN DE DND
SI
S9910
CODIFICAR EN ENTROPÍA LOS DATOS DIFERENCIALES QUE HAN PASADO A TRAVÉS DE LA OPERACIÓN DE DESPLAZAMIENTO HACIA ARRIBA
GENERAR FLUJO DE BITS

FIG. 10C

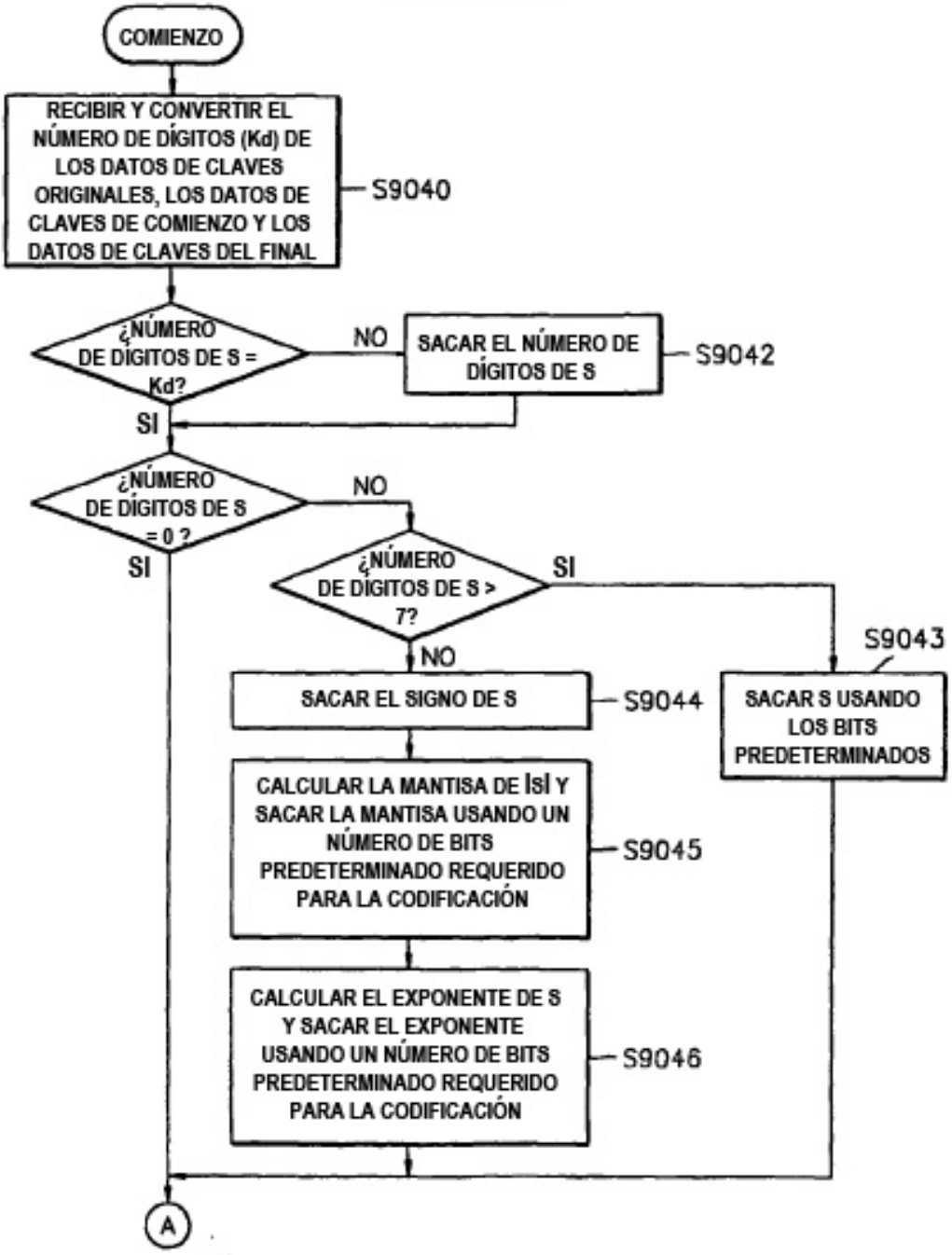
COMIENZO
RECIBIR Y CONVERTIR EL NÚMERO DE DÍGITOS (Kd) DE LOS DATOS DE CLAVES ORIGINALES, LOS DATOS DE CLAVES DE COMIENZO Y LOS DATOS DE CLAVES DEL FINAL
S9040
¿NÚMERO DE DIGITOS DE S = Kd?
NO
SACAR EL NÚMERO DE DÍGITOS DE S
S9042
SI
¿NÚMERO DE DÍGITOS DE S = 0 ?
NO
SI
¿NÚMERO DE DÍGITOS DE S > 7?
SI
NO
S9043
SACAR EL SIGNO DE S
S9044
SACAR S USANDO LOS BITS PREDETERMINADOS
CALCULAR LA MANTISA DE ISI Y SACAR LA MANTISA USANDO UN NÚMERO DE BITS PREDETERMINADO REQUERIDO PARA LA CODIFICACIÓN
S9045
CALCULAR EL EXPONENTE DE S Y SACAR EL EXPONENTE USANDO UN NÚMERO DE BITS PREDETERMINADO REQUERIDO PARA LA CODIFICACIÓN
S9046
A

FIG. 10D

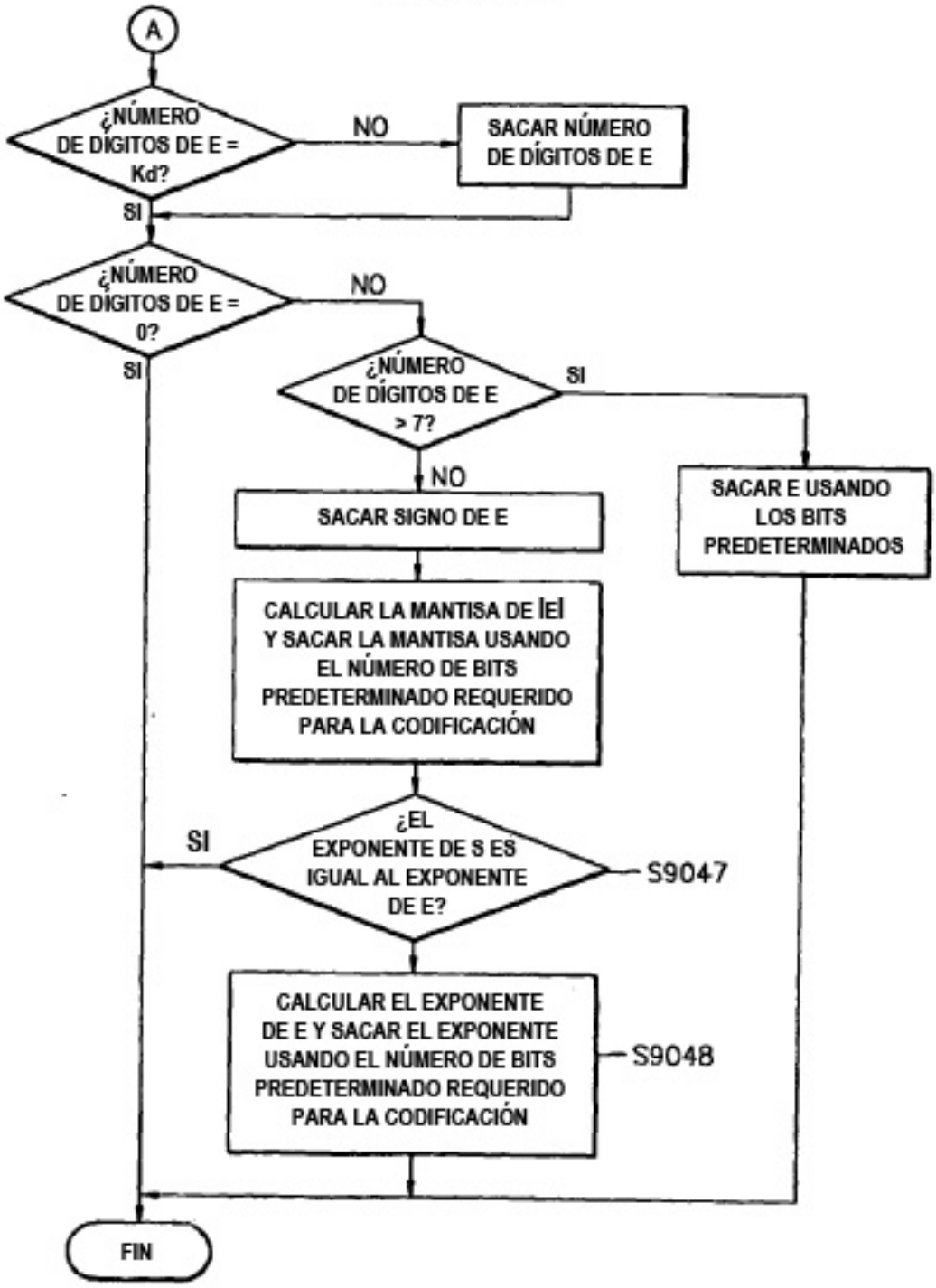
A
¿NÚMERO DE DÍGITOS DE E = Kd?
NO
SACAR NÚMERO DE DÍGITOS DE E
SI
¿NÚMERO DE DÍGITOS DE E = 0?
NO
SI
¿NÚMERO DE DÍGITOS DE E > 7?
SI
NO
SACAR SIGNO DE E
SACAR E USANDO LOS BITS PREDETERMINADOS
CALCULAR LA MANTISA DE |E| Y SACAR LA MANTISA USANDO EL NÚMERO DE BITS PREDETERMINADO REQUERIDO PARA LA CODIFICACIÓN
¿EL EXPONENTE DE S ES IGUAL AL EXPONENTE DE E?
SI
S9047
CALCULAR EL EXPONENTE DE E Y SACAR EL EXPONENTE USANDO EL NÚMERO DE BITS PREDETERMINADO REQUERIDO PARA LA CODIFICACIÓN
S9048
FIN

FIG. 10E

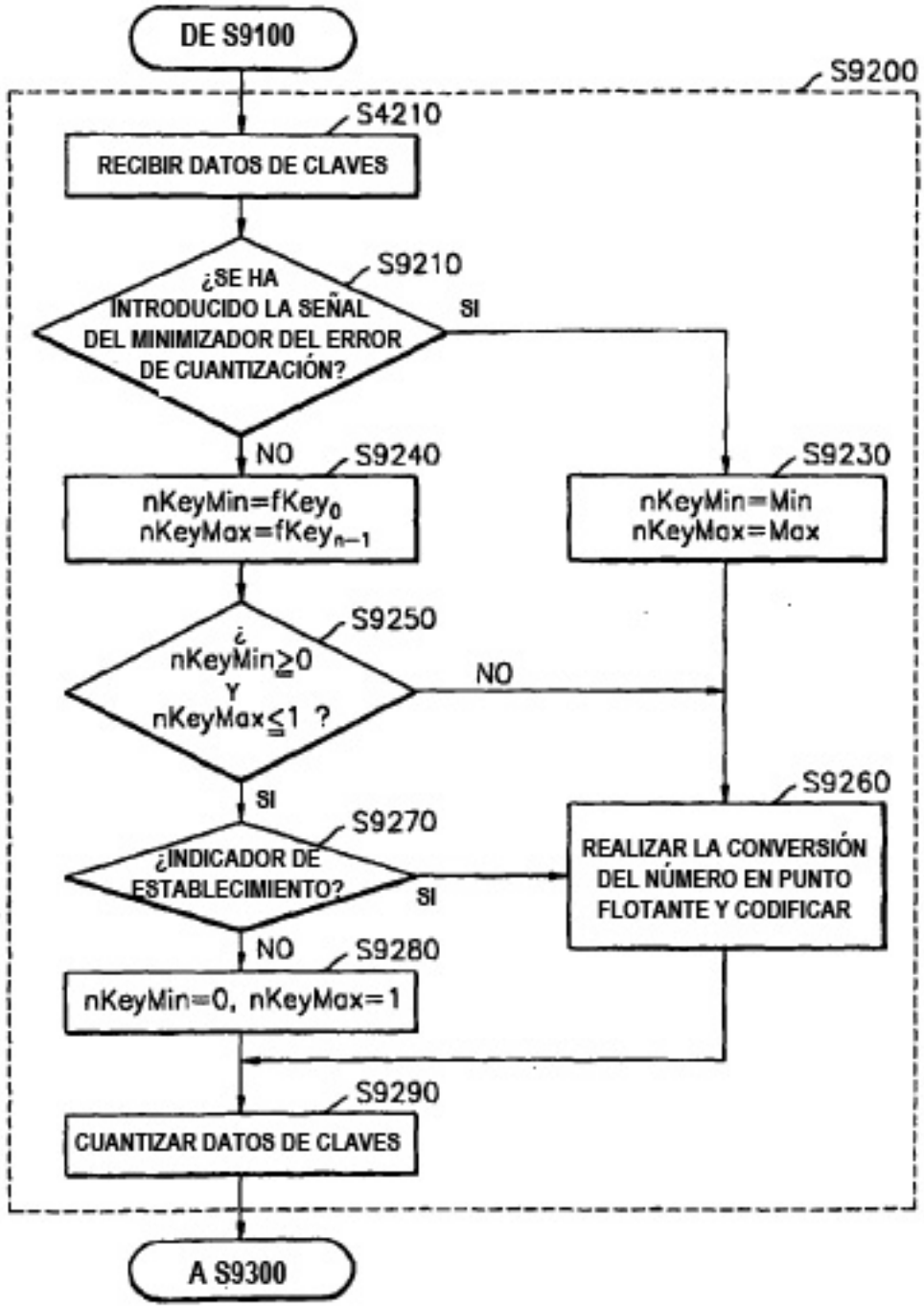
DE S9100
S9200
S4210
RECIBIR DATOS DE CLAVES
S9210
¿SE HA INTRODUCIDO LA SEÑAL DEL MINIMIZADOR DEL ERROR DE CUANTIZACIÓN?
SI
NO
S9240
nKeyMin=fKey0
nKeyMax=fKeyn-1
S9230
nKeyMin=Min
nKeyMax=Max
S9250
¿ nKeyMin≥0 Y nKeyMax≤1 ?
NO
SI
S9260
S9270
¿INDICADOR DE ESTABLECIMIENTO?
SI
REALIZAR LA CONVERSIÓN DEL NÚMERO EN PUNTO FLOTANTE Y CODIFICAR
NO
S9280
nKeyMin=0, nKeyMax=1
S9290
CUANTIZAR DATOS DE CLAVES
A S9300

FIG. 10F

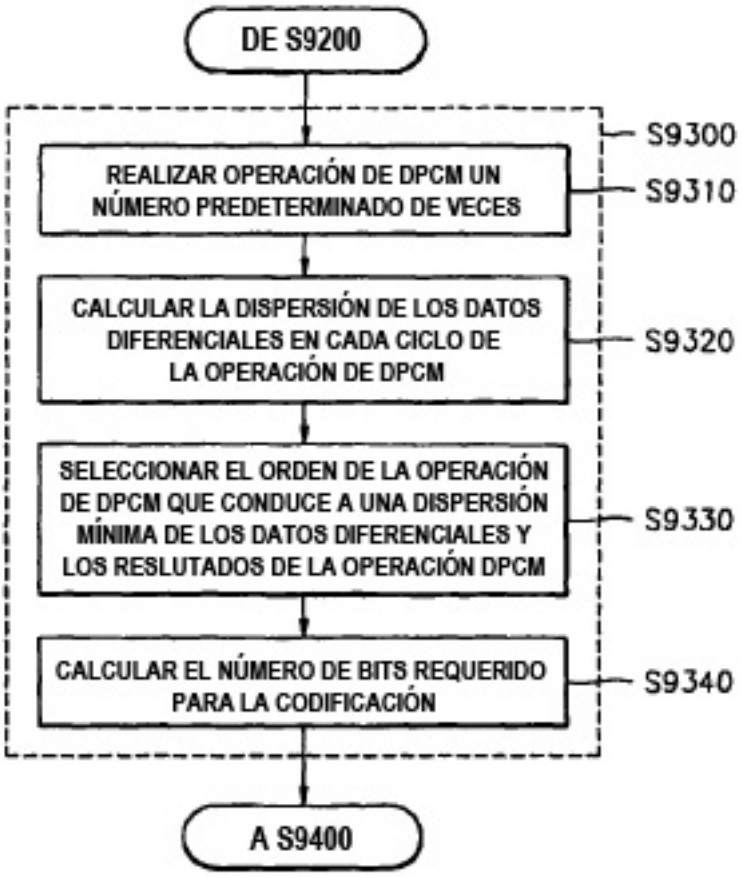
DE S9200
S9300
REALIZAR OPERACIÓN DE DPCM UN NÚMERO PREDETERMINADO DE VECES
S9310
CALCULAR LA DISPERSIÓN DE LOS DATOS DIFERENCIALES EN CADA CICLO DE LA OPERACIÓN DE DPCM
S9320
SELECCIONAR EL ORDEN DE LA OPERACIÓN DE DPCM QUE CONDUCE A UNA DISPERSIÓN MÍNIMA DE LOS DATOS DIFERENCIALES Y LOS RESLUTADOS DE LA OPERACIÓN DPCM
S9330
CALCULAR EL NÚMERO DE BITS REQUERIDO PARA LA CODIFICACIÓN
S9340
A S9400

FIG. 10G

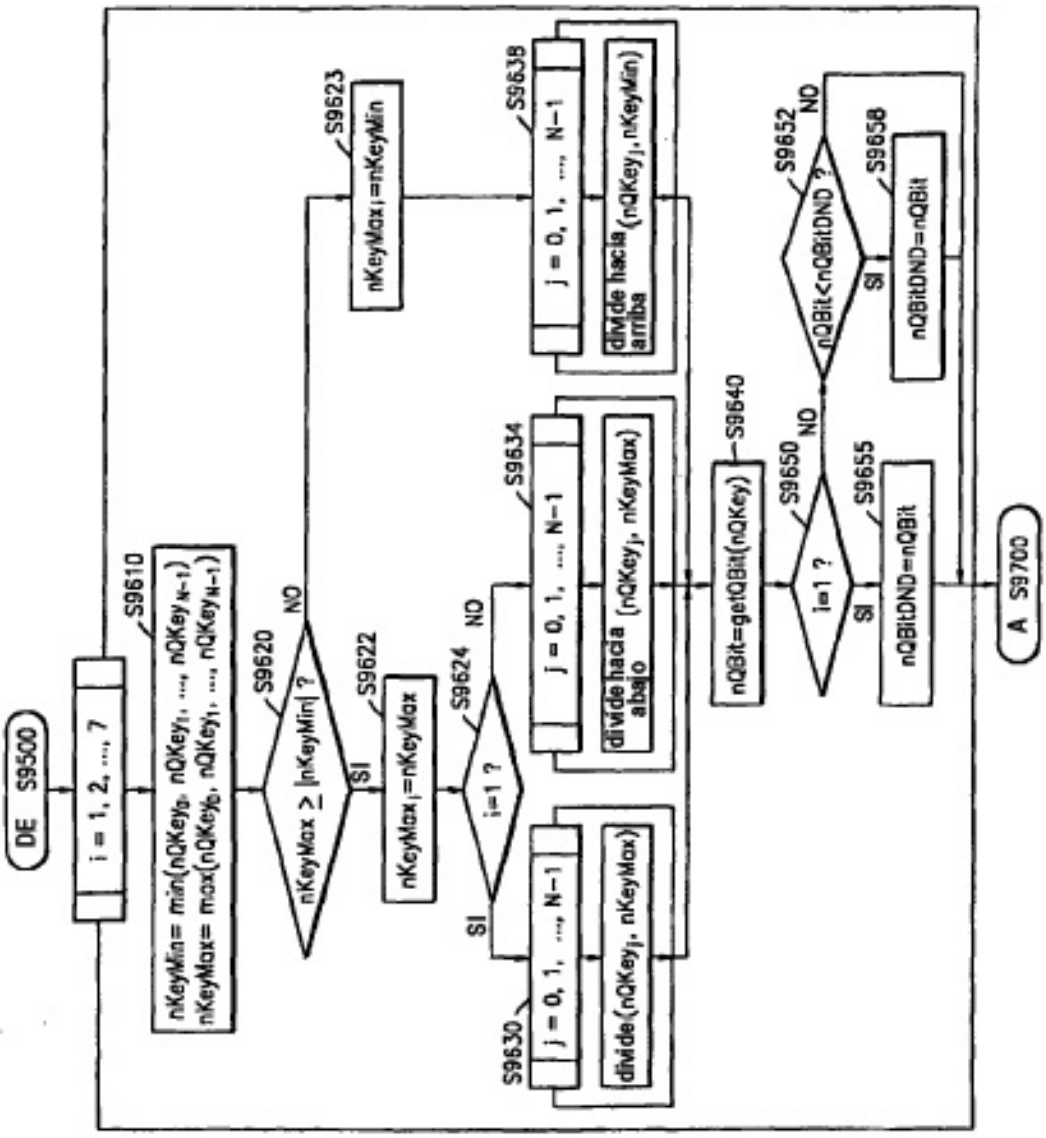
DE S9500
i = 1, 2, ..., 7
S9610
nKeyMin= min(nQKey0, nQKey1, ..., nQKeyN-1)
nKeyMax= max(nQKey0, nQKey1, ..., nQKeyN-1)
S9620
nKeyMax ≥ |nKeyMin| ?
NO
SI
S9622
nKeyMaxi=nKeyMax
S9623
nKeyMaxi=nKeyMin
S9624
i=1 ?
NO
SI
S9630
j = 0, 1, ..., N-1
divide(nQKeyj, nKeyMax)
S9634
j = 0, 1, ..., N-1
divide hacia abajo (nQKeyj, nKeyMax)
S9638
j = 0, 1, ..., N-1
divide hacia arriba (nQKeyj, nKeyMin)
S9640
nQBit=getQBit(nQKey)
S9650
i=1 ?
NO
SI
S9652
nQBit<nQBitDND ?
NO
SI
S9655
nQBitDND=nQBit
S9658
nQBitDND=nQBit
A S9700

FIG. 11

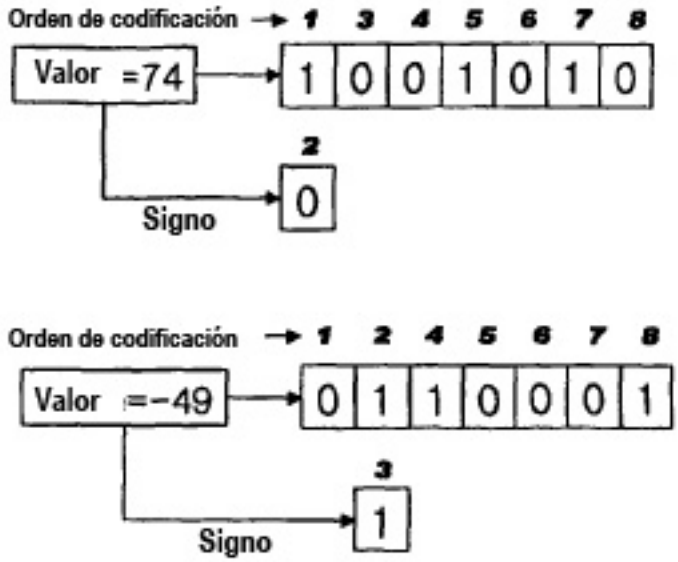
Orden de codificación → 1 3 4 5 6 7 8
Valor =74
1 0 0 1 0 1 0
2
0
Signo
Orden de codificación → 1 2 4 5 6 7 8
Valor =-49
0 1 1 0 0 0 1
3
1
Signo

FIG. 12A

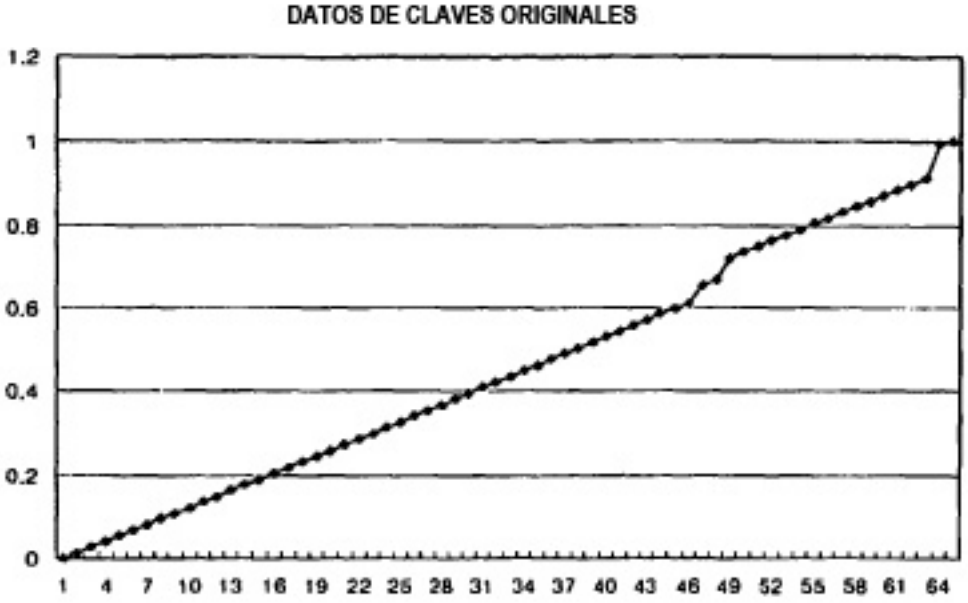

FIG. 12B

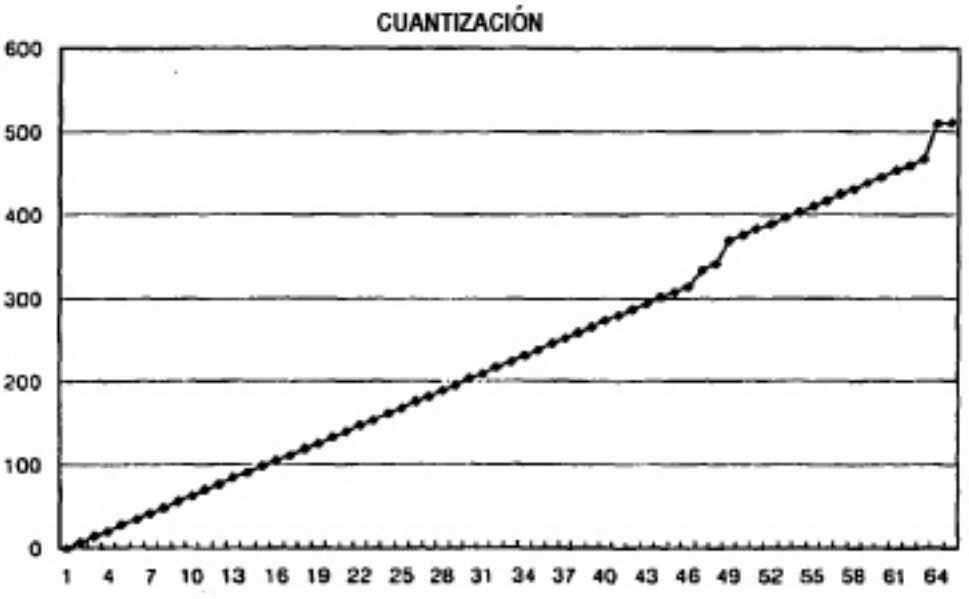

FIG. 12C

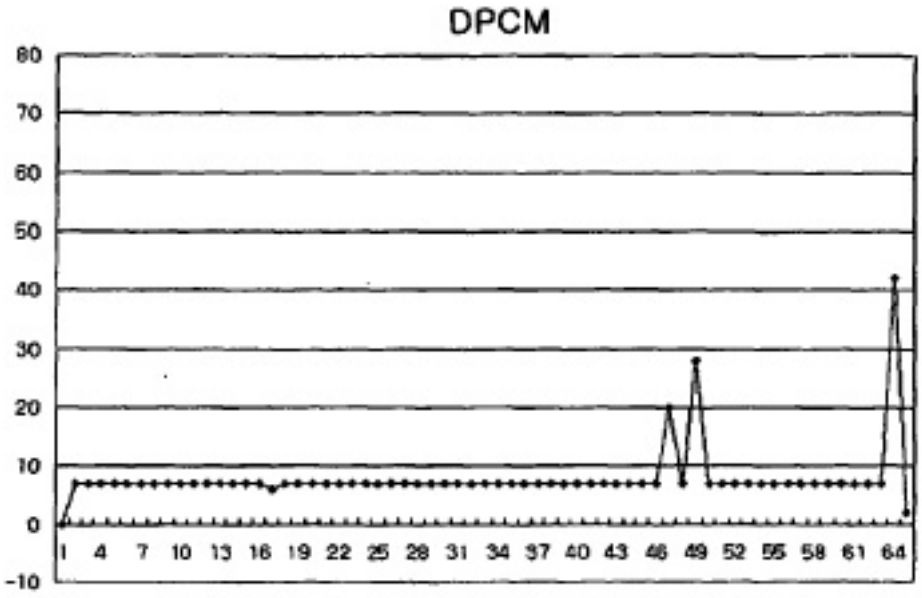

FIG. 12D

FIG. 12E

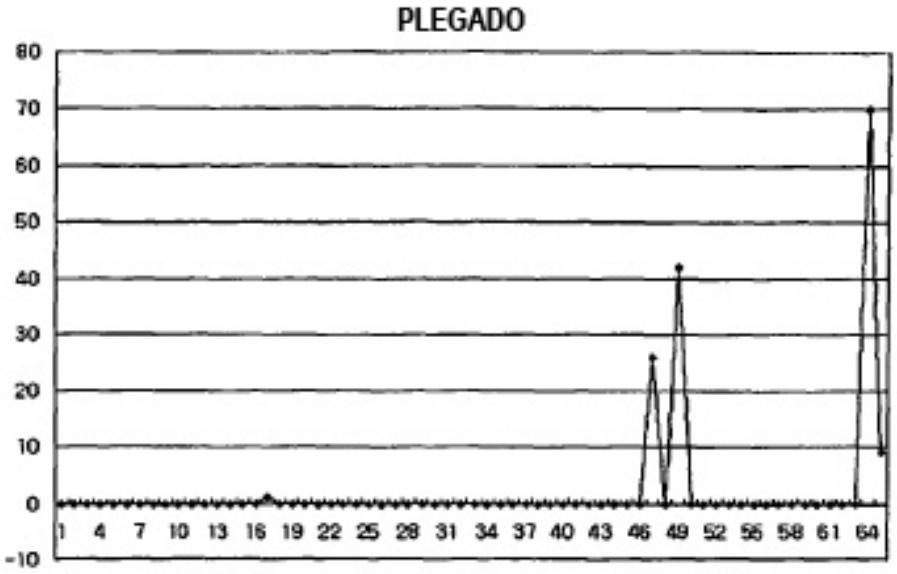

FIG. 12F

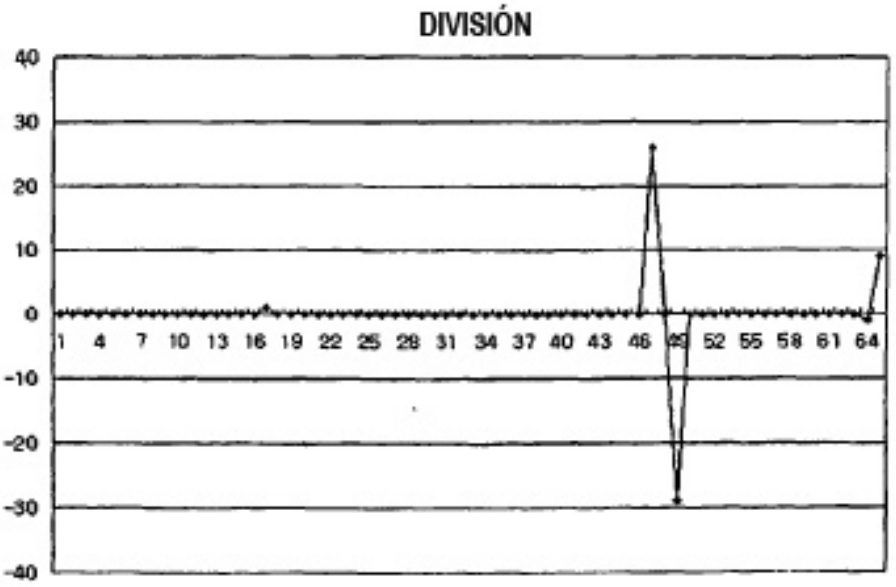

FIG. 12G

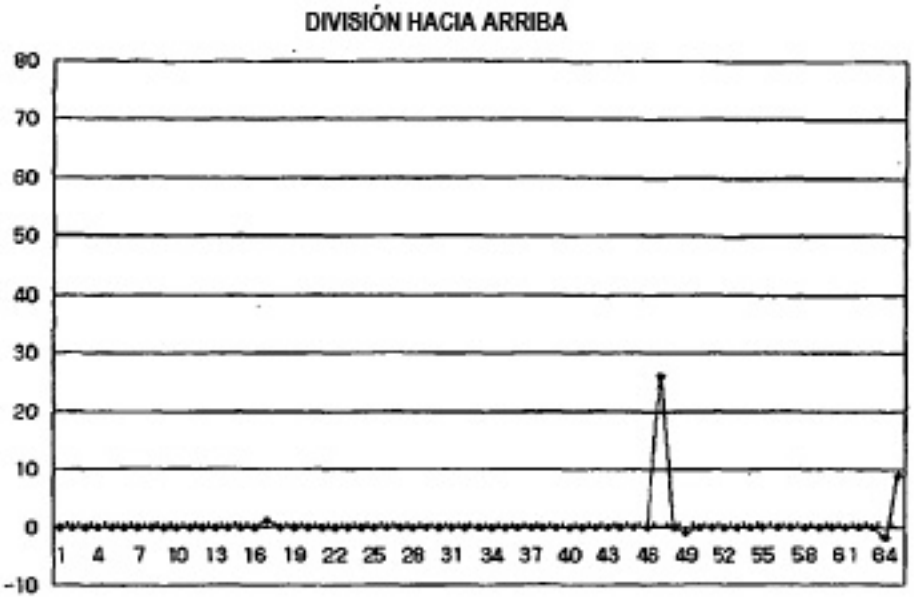

FIG. 12H

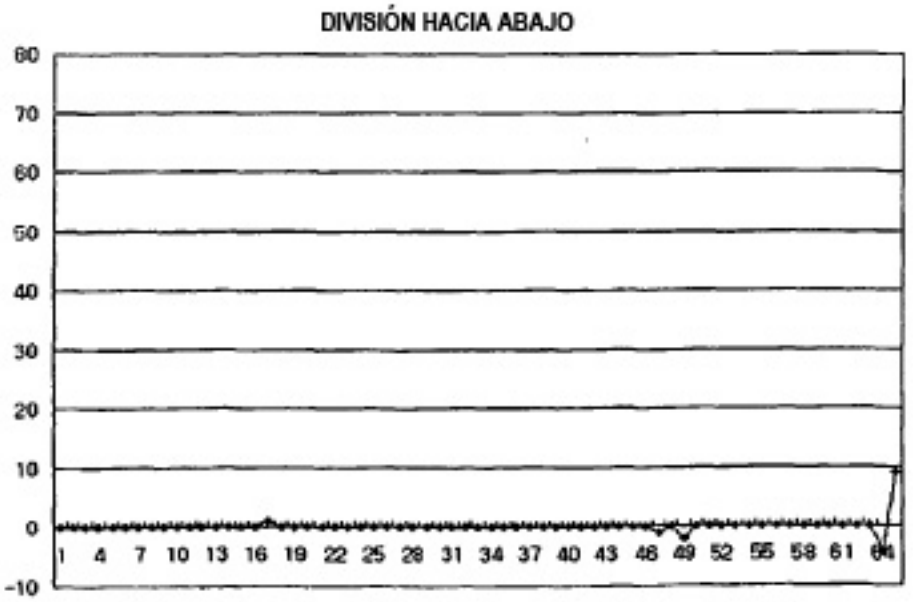

FIG. 12I

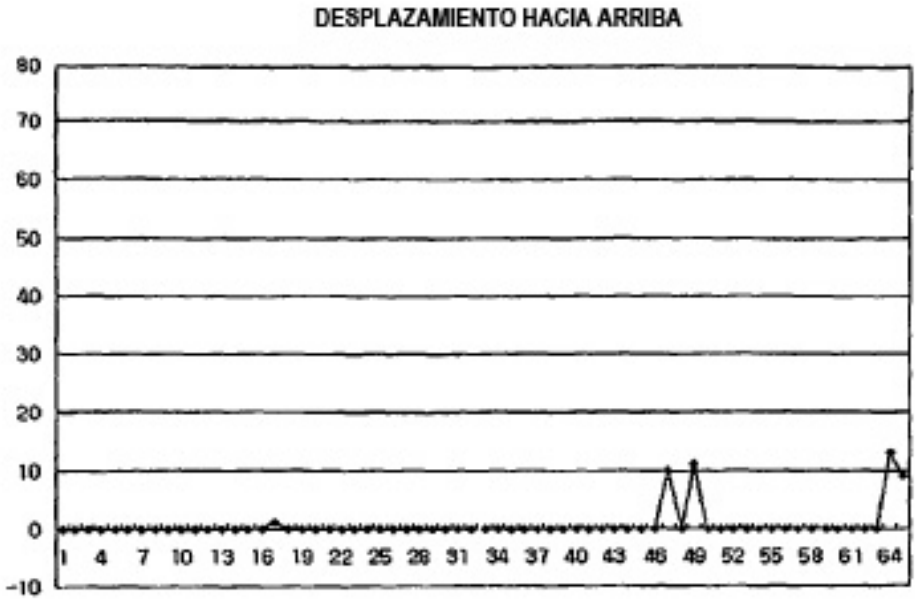

FIG. 12J

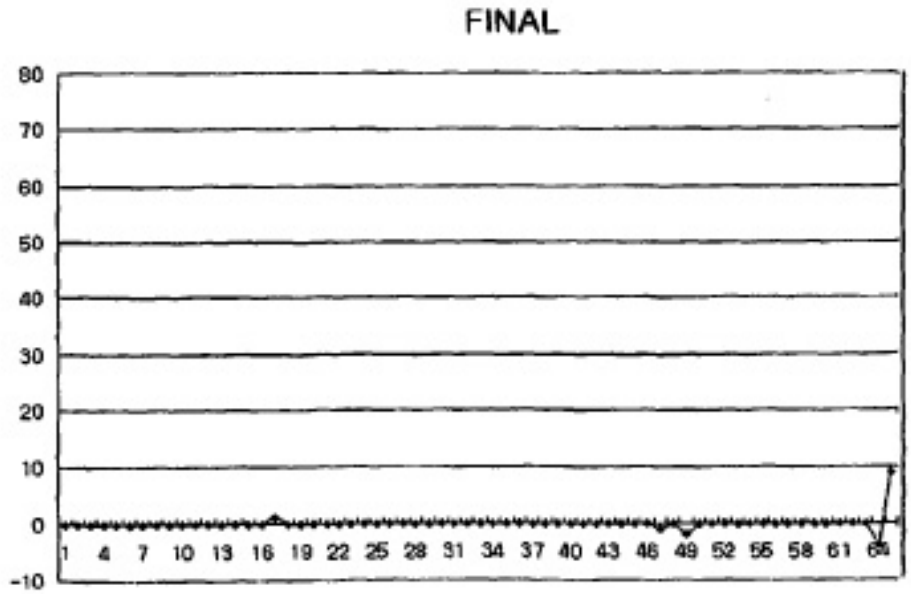

FIG. 13A

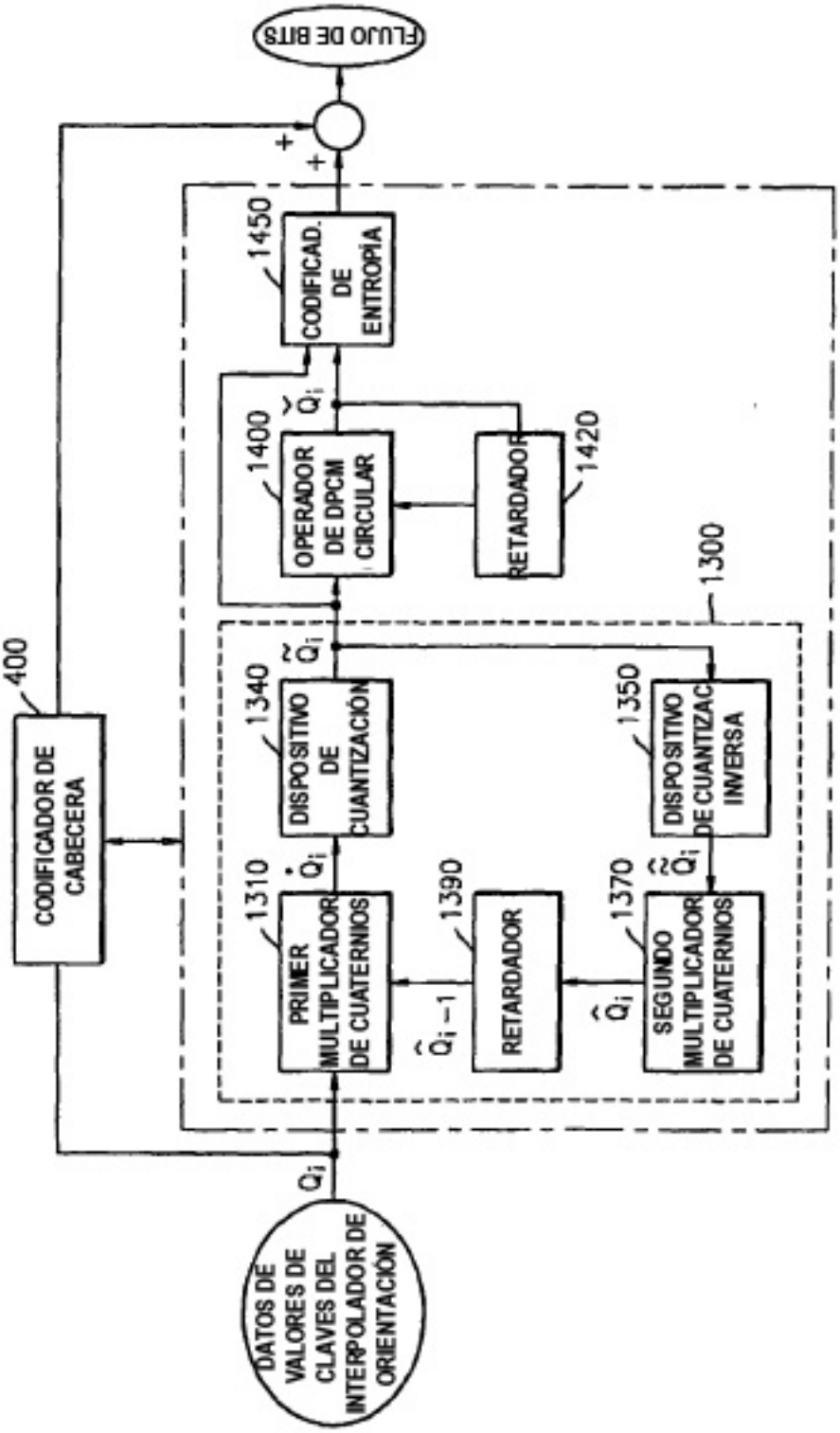
FLUJO DE BITS
+
+
1450
CODIFICAD. DE ENTROPÍA
1400
OPERADOR DE DPCM CIRCULAR
1420
RETARDADOR
1300
400
CODIFICADOR DE CABECERA
1340
DISPOSITIVO DE CUANTIZACIÓN
1350
DISPOSITIVO DE CUANTIZAC INVERSA
1310
PRIMER MULTIPLICADOR DE CUATERNIOS
1390
RETARDADOR
1370
SEGUNDO MULTIPLICADOR DE CUATERNIOS
DATOS DE VALORES DE CLAVES DEL INTERPOLADOR DE ORIENTACIÓN

FIG. 13B

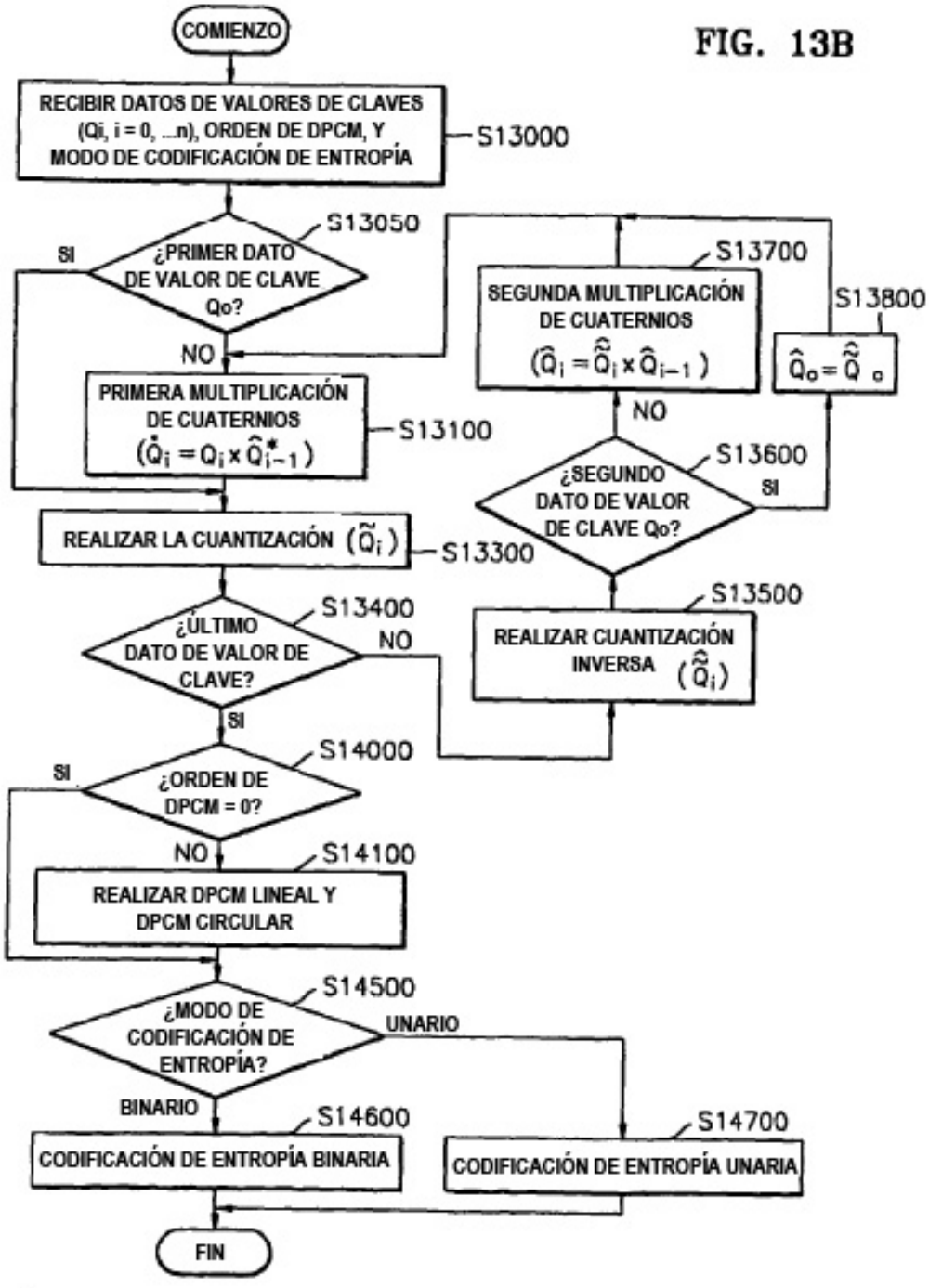
COMIENZO
RECIBIR DATOS DE VALORES DE CLAVES (Qi, i = 0, ...n), ORDEN DE DPCM, Y MODO DE CODIFICACIÓN DE ENTROPÍA
S13000
S13050
¿PRIMER DATO DE VALOR DE CLAVE Qo?
SI
NO
PRIMERA MULTIPLICACIÓN DE CUATERNIOS
S13100
REALIZAR LA CUANTIZACIÓN
S13300
S13400
¿ÚLTIMO DATO DE VALOR DE CLAVE?
NO
SI
S14000
¿ORDEN DE DPCM = 0?
SI
NO
S14100
REALIZAR DPCM LINEAL Y DPCM CIRCULAR
S14500
¿MODO DE CODIFICACIÓN DE ENTROPÍA?
UNARIO
BINARIO
S14600
CODIFICACIÓN DE ENTROPÍA BINARIA
S14700
CODIFICACIÓN DE ENTROPÍA UNARIA
FIN
S13700
SEGUNDA MULTIPLICACIÓN DE CUATERNIOS
S13800
NO
S13600
¿SEGUNDO DATO DE VALOR DE CLAVE Qo?
SI
S13500
REALIZAR CUANTIZACIÓN INVERSA

FIG. 14A

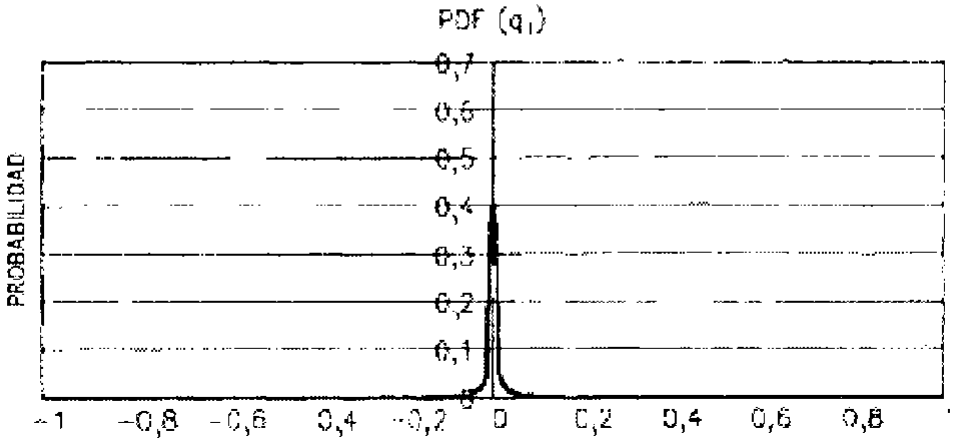

FIG. 14B

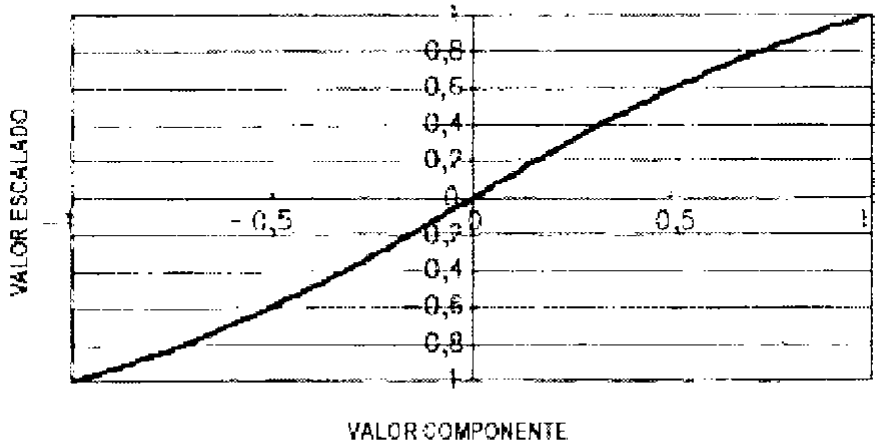

FIG. 15A

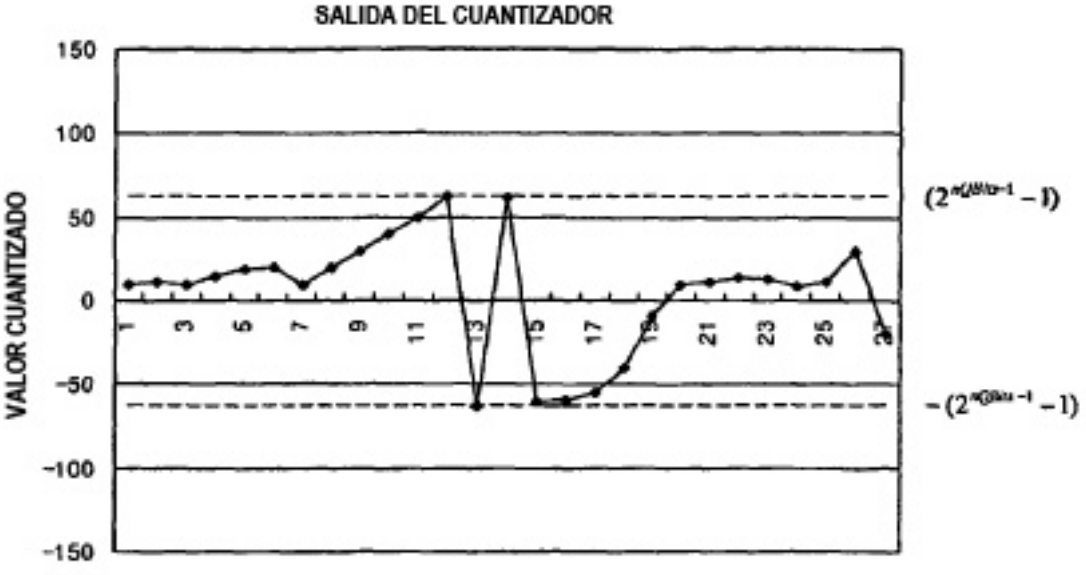

FIG. 15B

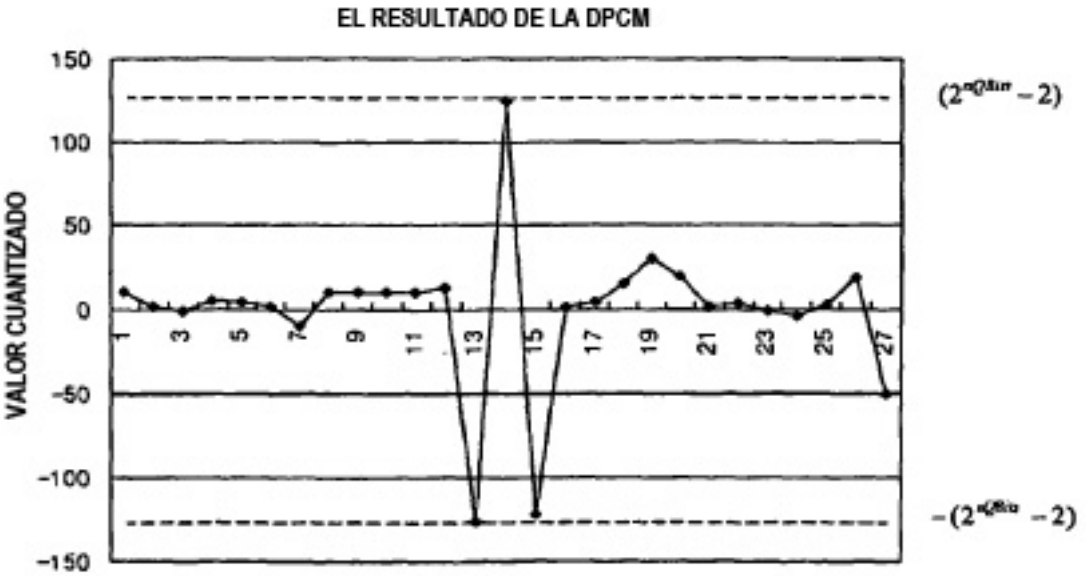

FIG. 15C

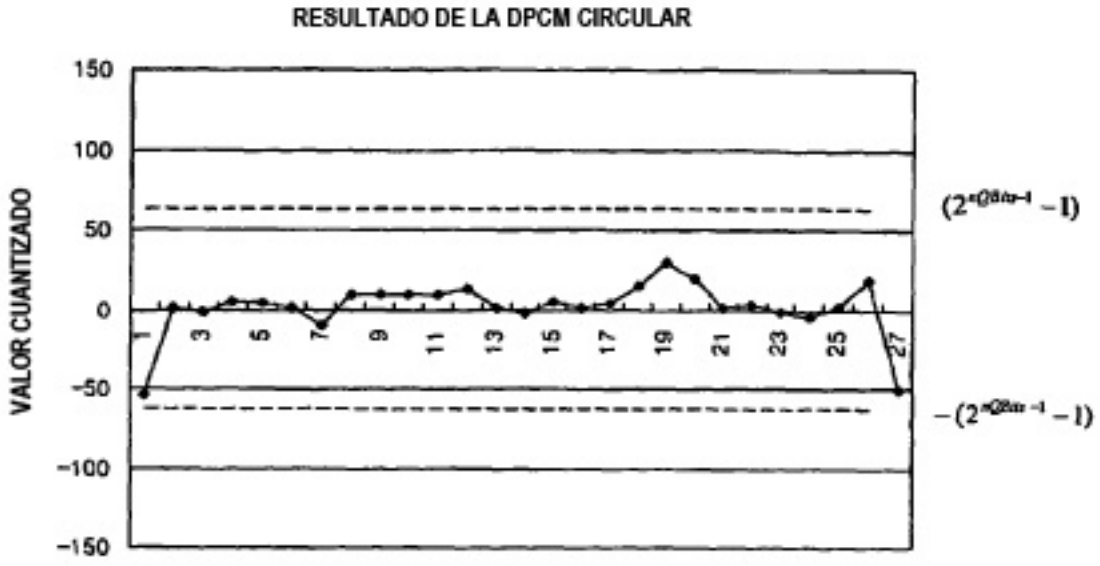
RESULTADO DE LA DPCM CIRCULAR
VALOR CUANTIZADO
150
100
50
0
-50
-100
-150
1
3
5
7
9
11
13
15
17
19
21
23
25
27
$(2^{n_{QBits}-1}-1)$
$-(2^{n_{QBits}-1}-1)$

FIG. 16

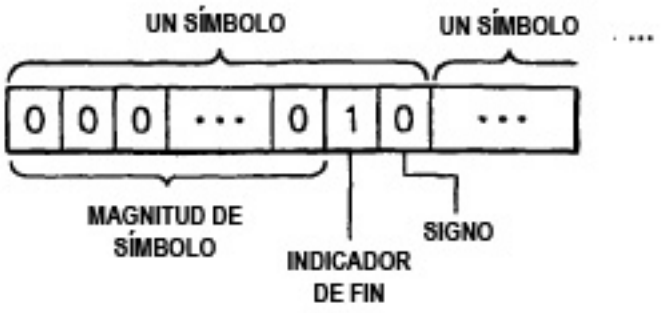
UN SÍMBOLO
UN SÍMBOLO
0 0 0 ··· 0 1 0 ···
MAGNITUD DE SÍMBOLO
INDICADOR DE FIN
SIGNO

FIG. 17

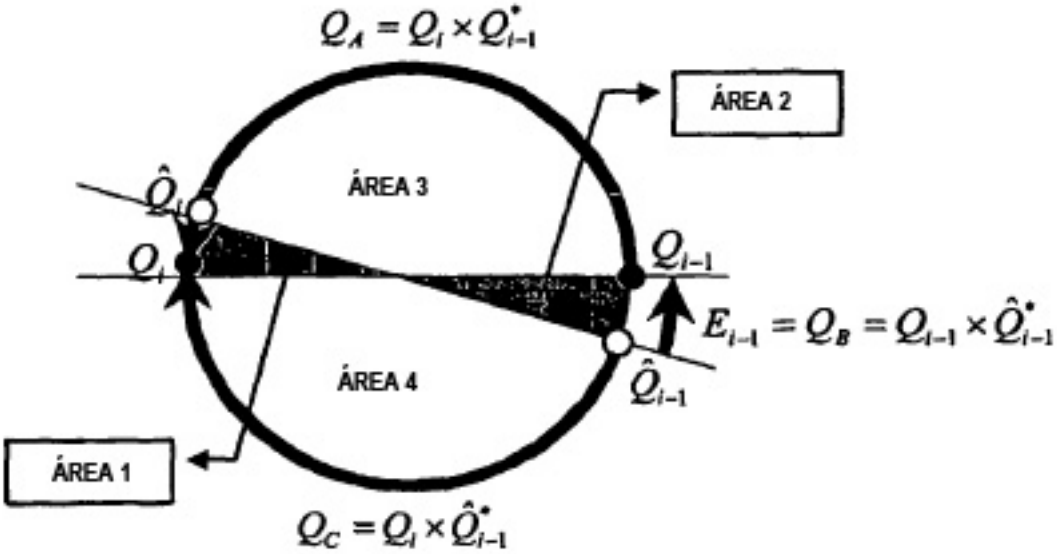
$Q_A = Q_l \times Q^*_{l-1}$
ÁREA 2
ÁREA 3
$\hat{Q}_l$
Q_l
Q_{l-1}
$E_{l-1} = Q_B = Q_{l-1} \times \hat{Q}^*_{l-1}$
ÁREA 4
$\hat{Q}_{l-1}$
ÁREA 1
$Q_C = Q_l \times \hat{Q}^*_{l-1}$

FIG. 18A

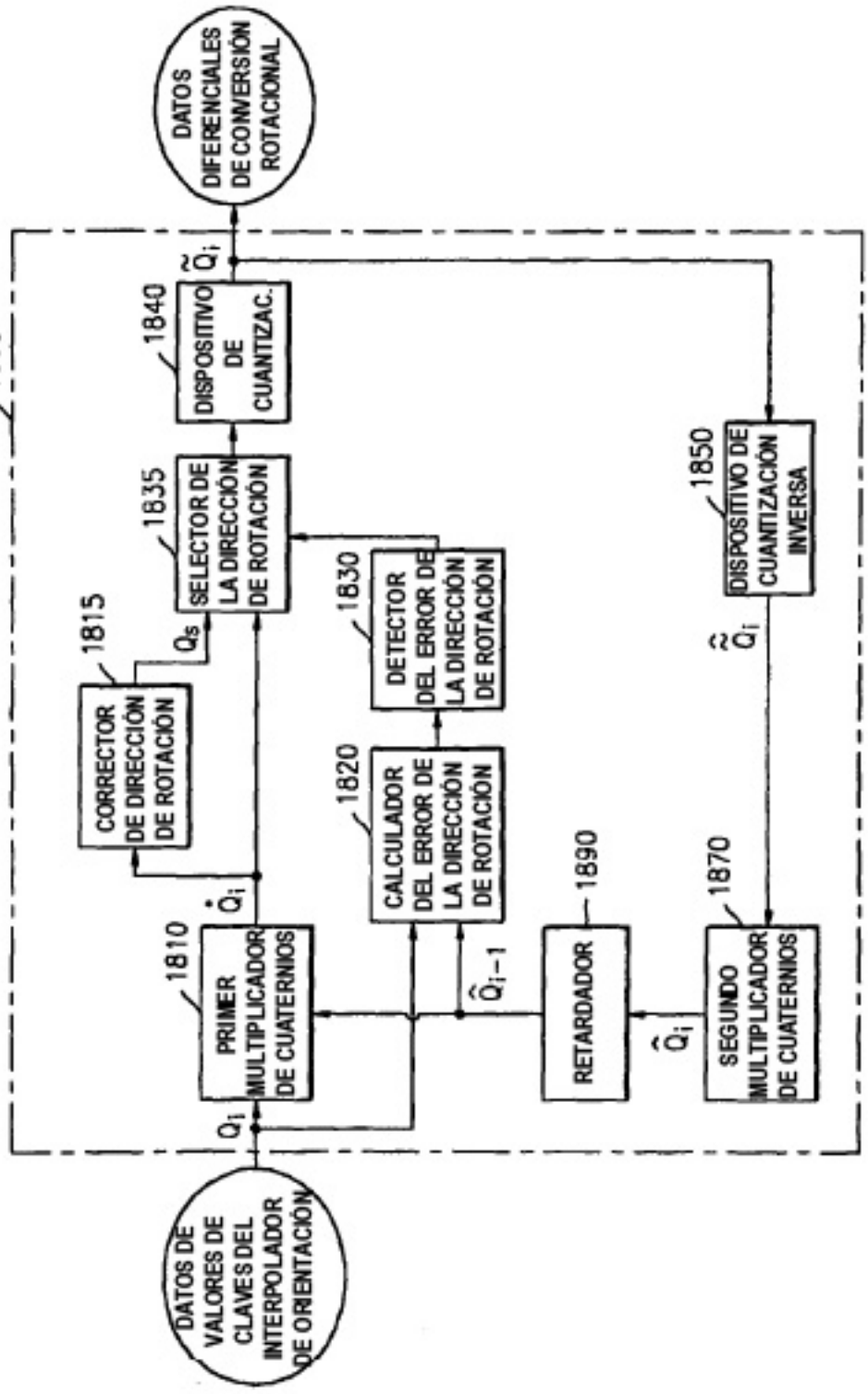
1800
DATOS DIFERENCIALES DE CONVERSIÓN ROTACIONAL
$\tilde{Q}_i$
1840
DISPOSITIVO DE CUANTIZAC.
1835
SELECTOR DE LA DIRECCIÓN DE ROTACIÓN
1815
CORRECTOR DE DIRECCIÓN DE ROTACIÓN
Q_s
1830
DETECTOR DEL ERROR DE LA DIRECCIÓN DE ROTACIÓN
1820
CALCULADOR DEL ERROR DE LA DIRECCIÓN DE ROTACIÓN
$\dot{Q}_i$
1810
PRIMER MULTIPLICADOR DE CUATERNIOS
Q_i
DATOS DE VALORES DE CLAVES DEL INTERPOLADOR DE ORIENTACIÓN
$\hat{Q}_{i-1}$
1890
RETARDADOR
$\hat{Q}_i$
1870
SEGUNDO MULTIPLICADOR DE CUATERNIOS
1850
DISPOSITIVO DE CUANTIZACIÓN INVERSA
$\tilde{\tilde{Q}}_i$

FIG. 18B

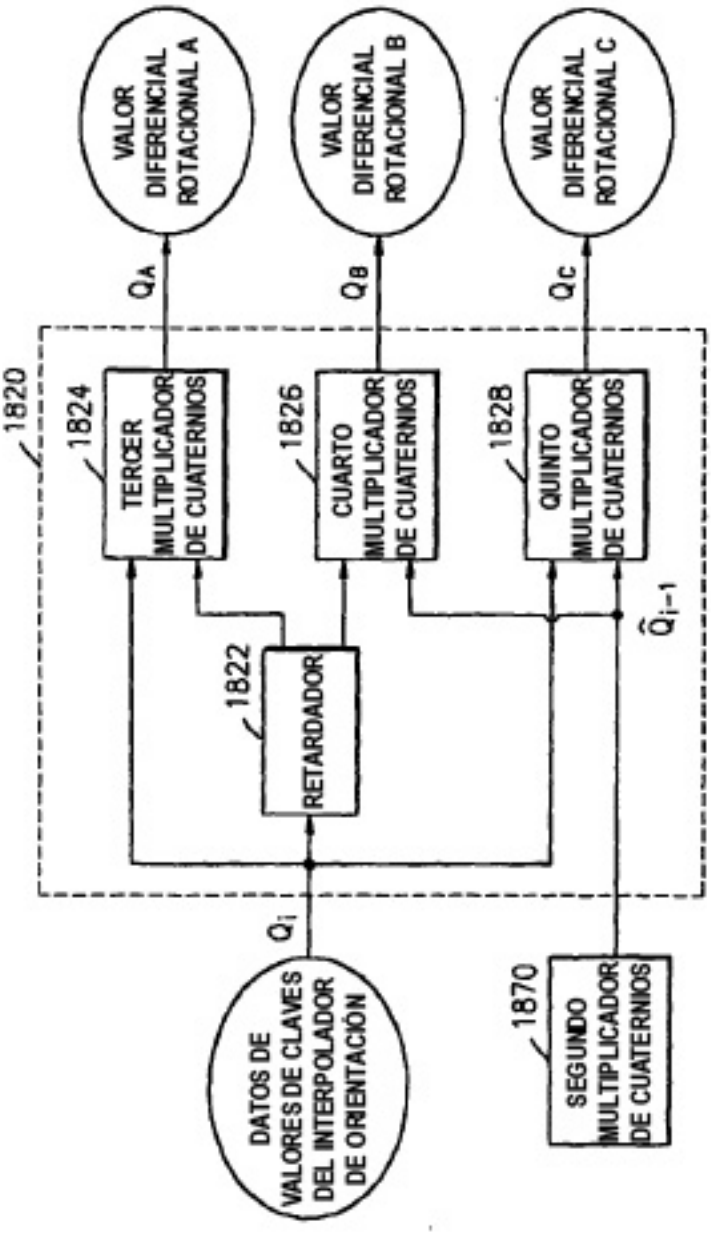
VALOR DIFERENCIAL ROTACIONAL A
VALOR DIFERENCIAL ROTACIONAL B
VALOR DIFERENCIAL ROTACIONAL C
Q_A
Q_B
Q_C
1820
1824
TERCER MULTIPLICADOR DE CUATERNIOS
1826
CUARTO MULTIPLICADOR DE CUATERNIOS
1828
QUINTO MULTIPLICADOR DE CUATERNIOS
1822
RETARDADOR
$\hat{Q}_{i-1}$
Q_i
DATOS DE VALORES DE CLAVES DEL INTERPOLADOR DE ORIENTACIÓN
1870
SEGUNDO MULTIPLICADOR DE CUATERNIOS

FIG. 19A

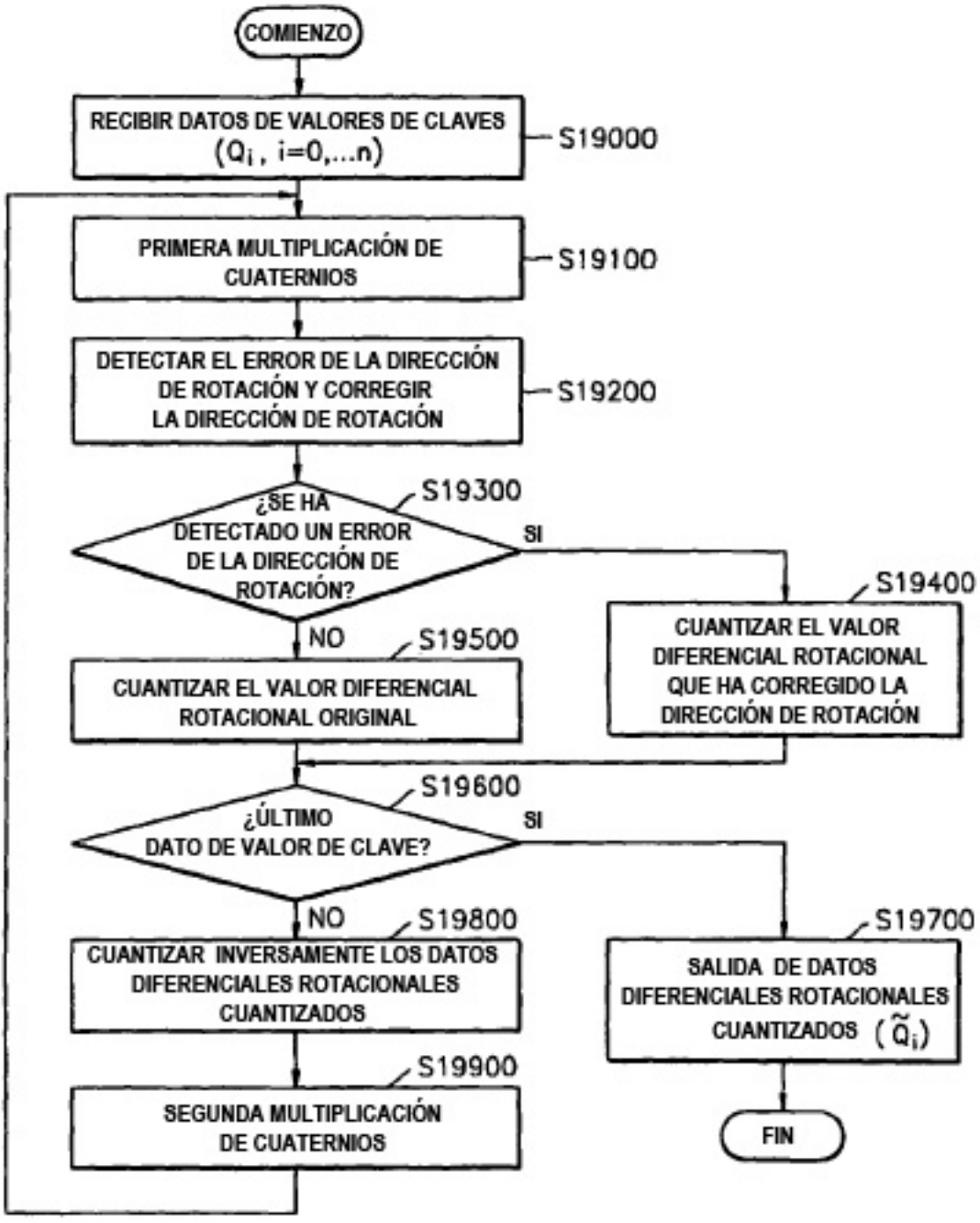
COMIENZO
RECIBIR DATOS DE VALORES DE CLAVES (Q_i, i=0,...n)
S19000
PRIMERA MULTIPLICACIÓN DE CUATERNIOS
S19100
DETECTAR EL ERROR DE LA DIRECCIÓN DE ROTACIÓN Y CORREGIR LA DIRECCIÓN DE ROTACIÓN
S19200
¿SE HA DETECTADO UN ERROR DE LA DIRECCIÓN DE ROTACIÓN?
S19300
SI
NO
CUANTIZAR EL VALOR DIFERENCIAL ROTACIONAL ORIGINAL
S19500
CUANTIZAR EL VALOR DIFERENCIAL ROTACIONAL QUE HA CORREGIDO LA DIRECCIÓN DE ROTACIÓN
S19400
¿ÚLTIMO DATO DE VALOR DE CLAVE?
S19600
SI
NO
CUANTIZAR INVERSAMENTE LOS DATOS DIFERENCIALES ROTACIONALES CUANTIZADOS
S19800
SALIDA DE DATOS DIFERENCIALES ROTACIONALES CUANTIZADOS ($\tilde{Q}_i$)
S19700
SEGUNDA MULTIPLICACIÓN DE CUATERNIOS
S19900
FIN

FIG. 19B

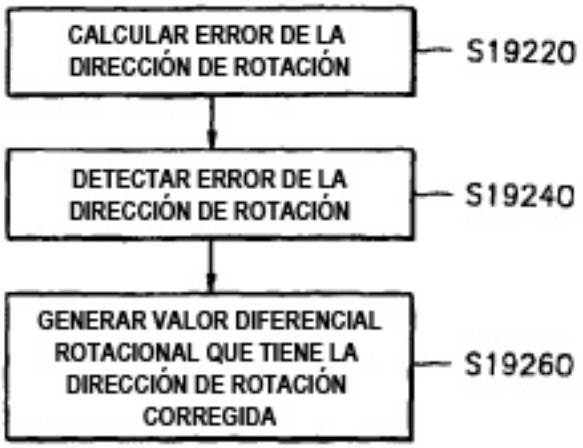
CALCULAR ERROR DE LA DIRECCIÓN DE ROTACIÓN
S19220
DETECTAR ERROR DE LA DIRECCIÓN DE ROTACIÓN
S19240
GENERAR VALOR DIFERENCIAL ROTACIONAL QUE TIENE LA DIRECCIÓN DE ROTACIÓN CORREGIDA
S19260

FIG. 20A

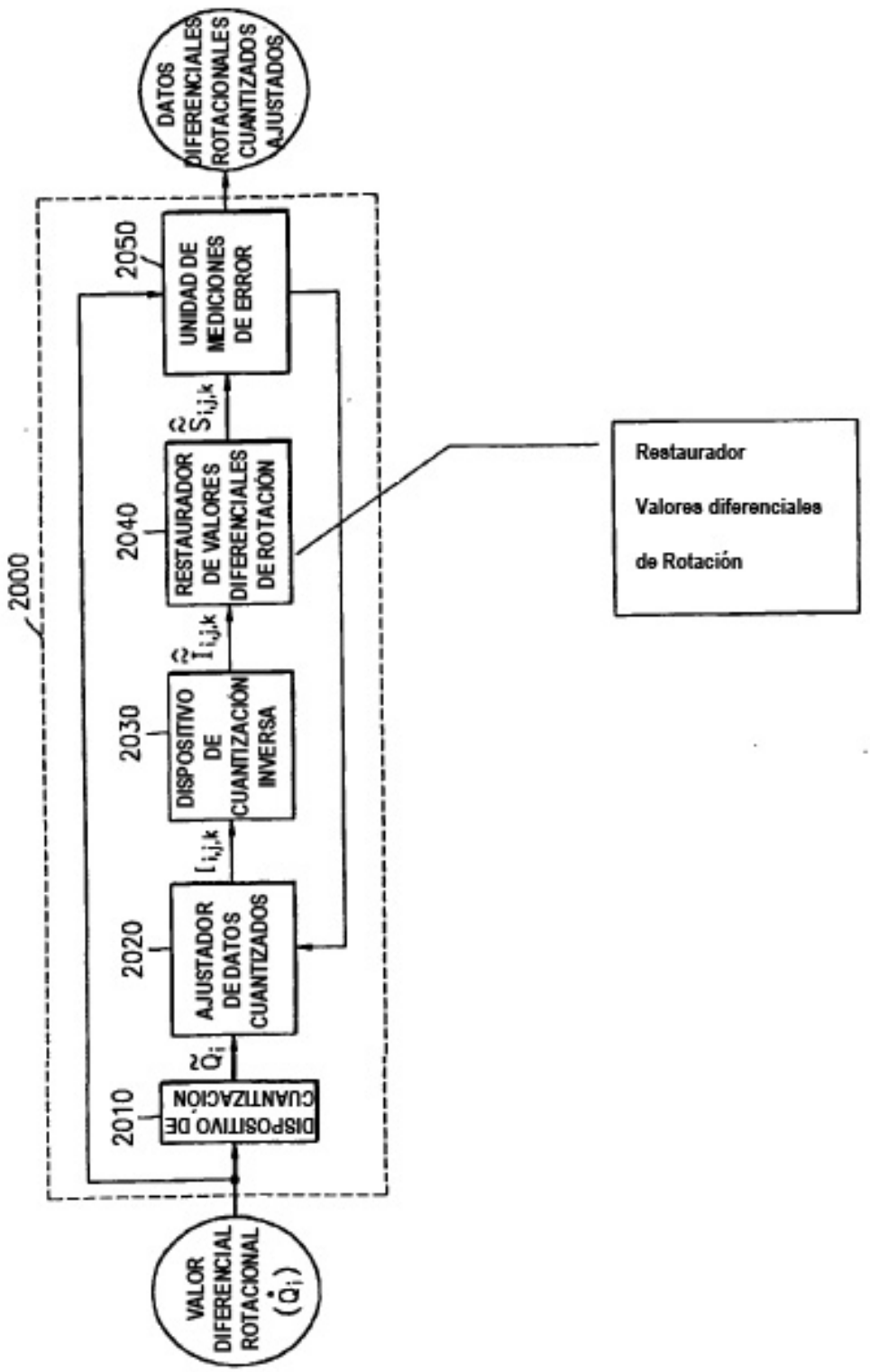
DATOS DIFERENCIALES ROTACIONALES CUANTIZADOS AJUSTADOS
2000
2050
UNIDAD DE MEDICIONES DE ERROR
2040
RESTAURADOR DE VALORES DIFERENCIALES DE ROTACIÓN
2030
DISPOSITIVO DE CUANTIZACIÓN INVERSA
2020
AJUSTADOR DE DATOS CUANTIZADOS
2010
DISPOSITIVO DE CUANTIZACIÓN
VALOR DIFERENCIAL ROTACIONAL
Restaurador
Valores diferenciales
de Rotación

FIG. 20B

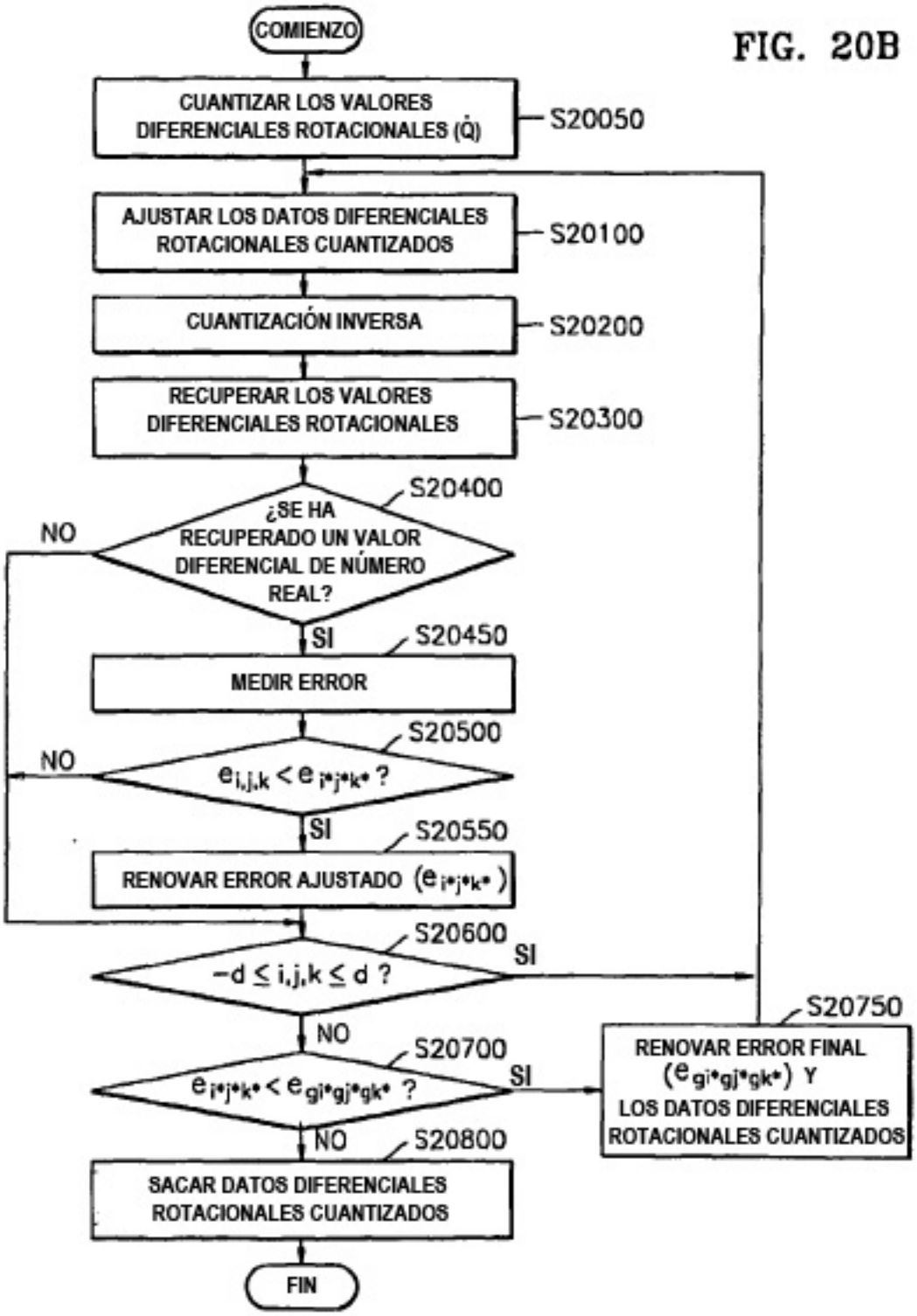
COMIENZO
CUANTIZAR LOS VALORES DIFERENCIALES ROTACIONALES (Q)
S20050
AJUSTAR LOS DATOS DIFERENCIALES ROTACIONALES CUANTIZADOS
S20100
CUANTIZACIÓN INVERSA
S20200
RECUPERAR LOS VALORES DIFERENCIALES ROTACIONALES
S20300
S20400
¿SE HA RECUPERADO UN VALOR DIFERENCIAL DE NÚMERO REAL?
NO
SI
S20450
MEDIR ERROR
S20500
$e_{i,j,k} < e_{i^*j^*k^*}$?
NO
SI
S20550
RENOVAR ERROR AJUSTADO ($e_{i^*j^*k^*}$)
S20600
$-d \leq i,j,k \leq d$?
SI
NO
S20700
$e_{i^*j^*k^*} < e_{gi^*gj^*gk^*}$?
SI
NO
S20750
RENOVAR ERROR FINAL ($e_{gi^*gj^*gk^*}$) Y LOS DATOS DIFERENCIALES ROTACIONALES CUANTIZADOS
S20800
SACAR DATOS DIFERENCIALES ROTACIONALES CUANTIZADOS
FIN

FIG. 21A

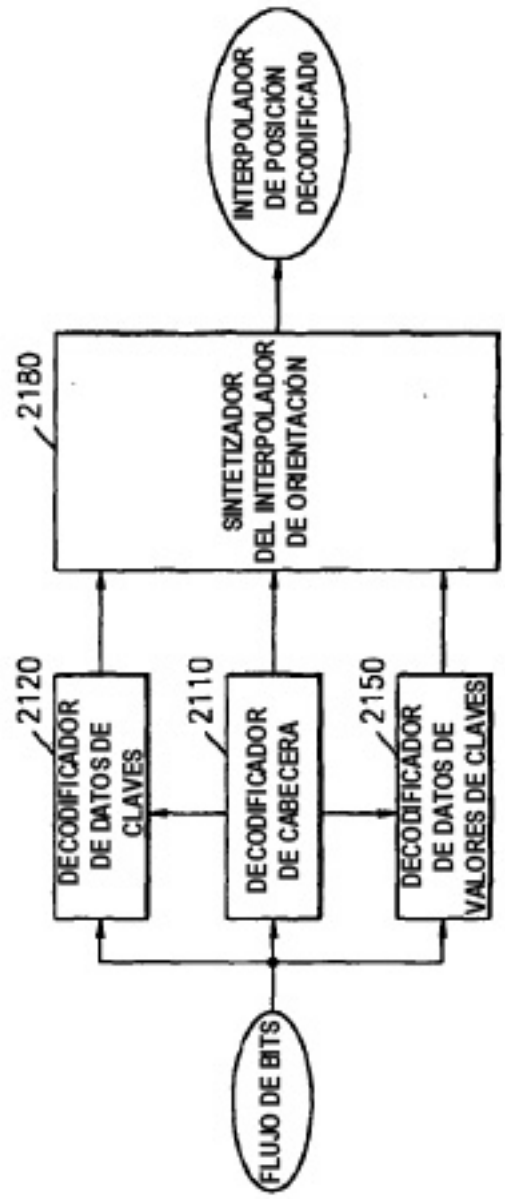
FLUJO DE BITS
2120
DECODIFICADOR DE DATOS DE CLAVES
2110
DECODIFICADOR DE CABECERA
2150
DECODIFICADOR DE DATOS DE VALORES DE CLAVES
2180
SINTETIZADOR DEL INTERPOLADOR DE ORIENTACIÓN
INTERPOLADOR DE POSICIÓN DECODIFICADO

FIG. 21B

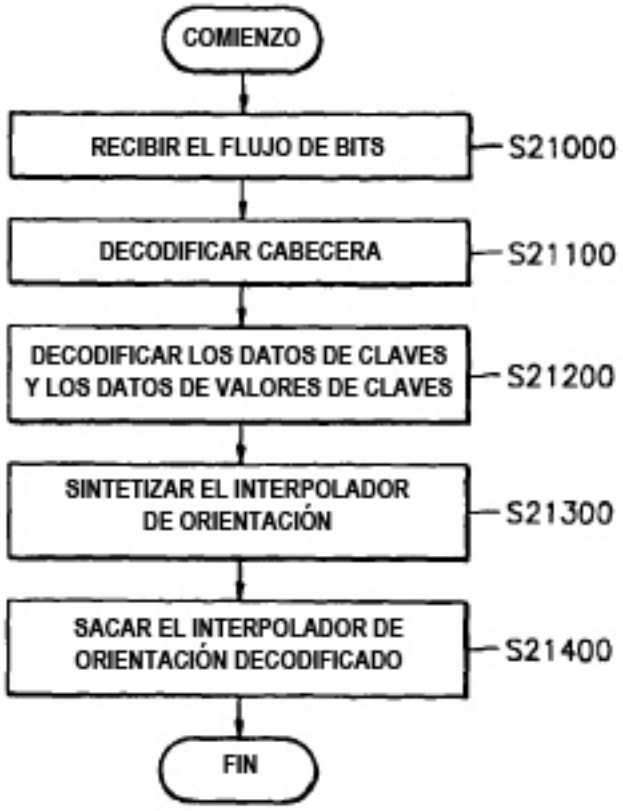
COMIENZO
RECIBIR EL FLUJO DE BITS
S21000
DECODIFICAR CABECERA
S21100
DECODIFICAR LOS DATOS DE CLAVES Y LOS DATOS DE VALORES DE CLAVES
S21200
SINTETIZAR EL INTERPOLADOR DE ORIENTACIÓN
S21300
SACAR EL INTERPOLADOR DE ORIENTACIÓN DECODIFICADO
S21400
FIN

FIG. 22

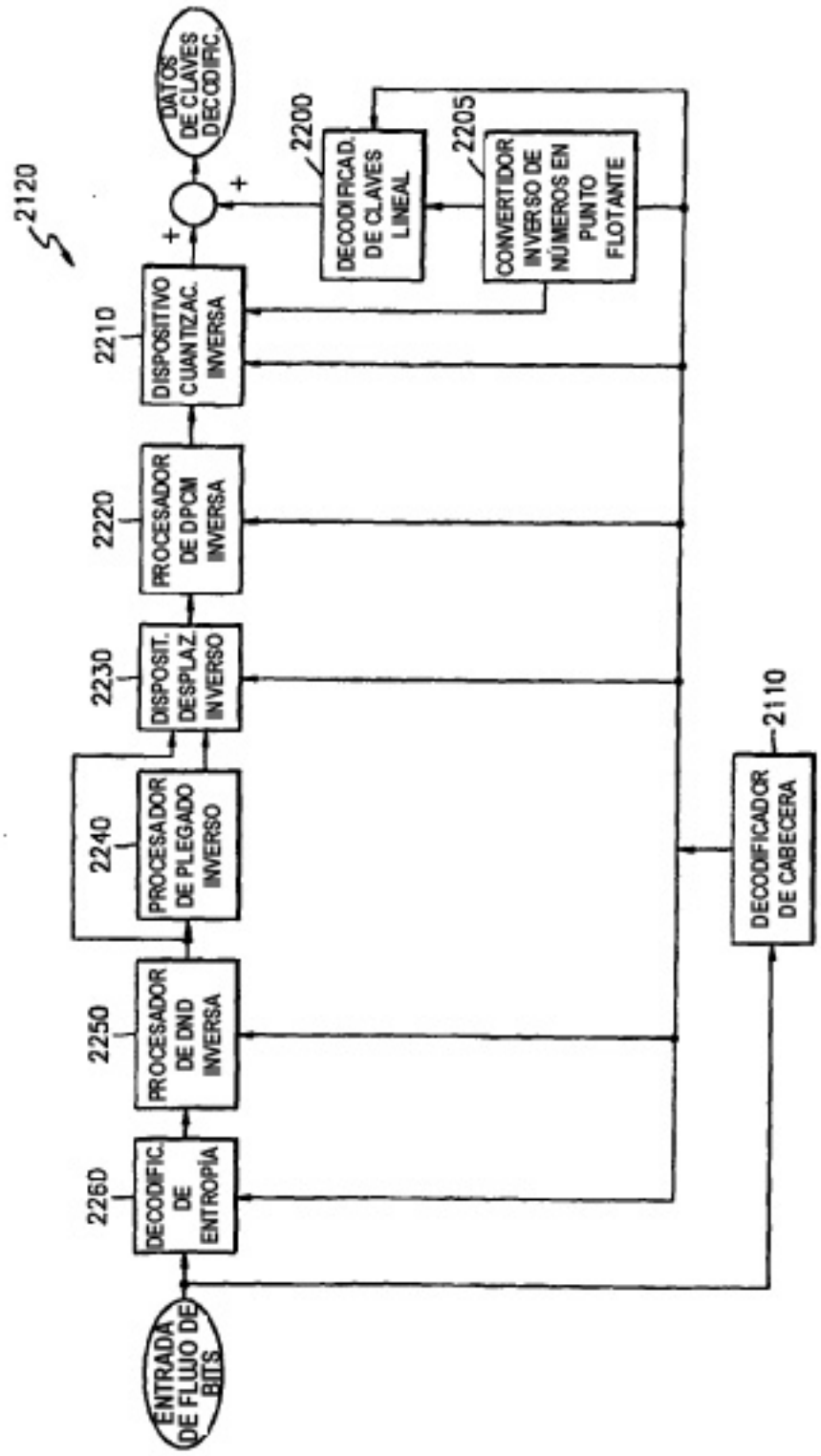
2120
DATOS DE CLAVES DECODIFIC.
2200
DECODIFICAD. DE CLAVES LINEAL
2205
CONVERTIDOR INVERSO DE NÚMEROS EN PUNTO FLOTANTE
2210
DISPOSITIVO CUANTIZAC. INVERSA
2220
PROCESADOR DE DPCM INVERSA
2230
DISPOSIT. DESPLAZ INVERSO
2240
PROCESADOR DE PLEGADO INVERSO
2250
PROCESADOR DE DWD INVERSA
2260
DECODIFIC. DE ENTROPÍA
ENTRADA DE FLUJO DE BITS
2110
DECODIFICADOR DE CABECERA

FIG. 23A

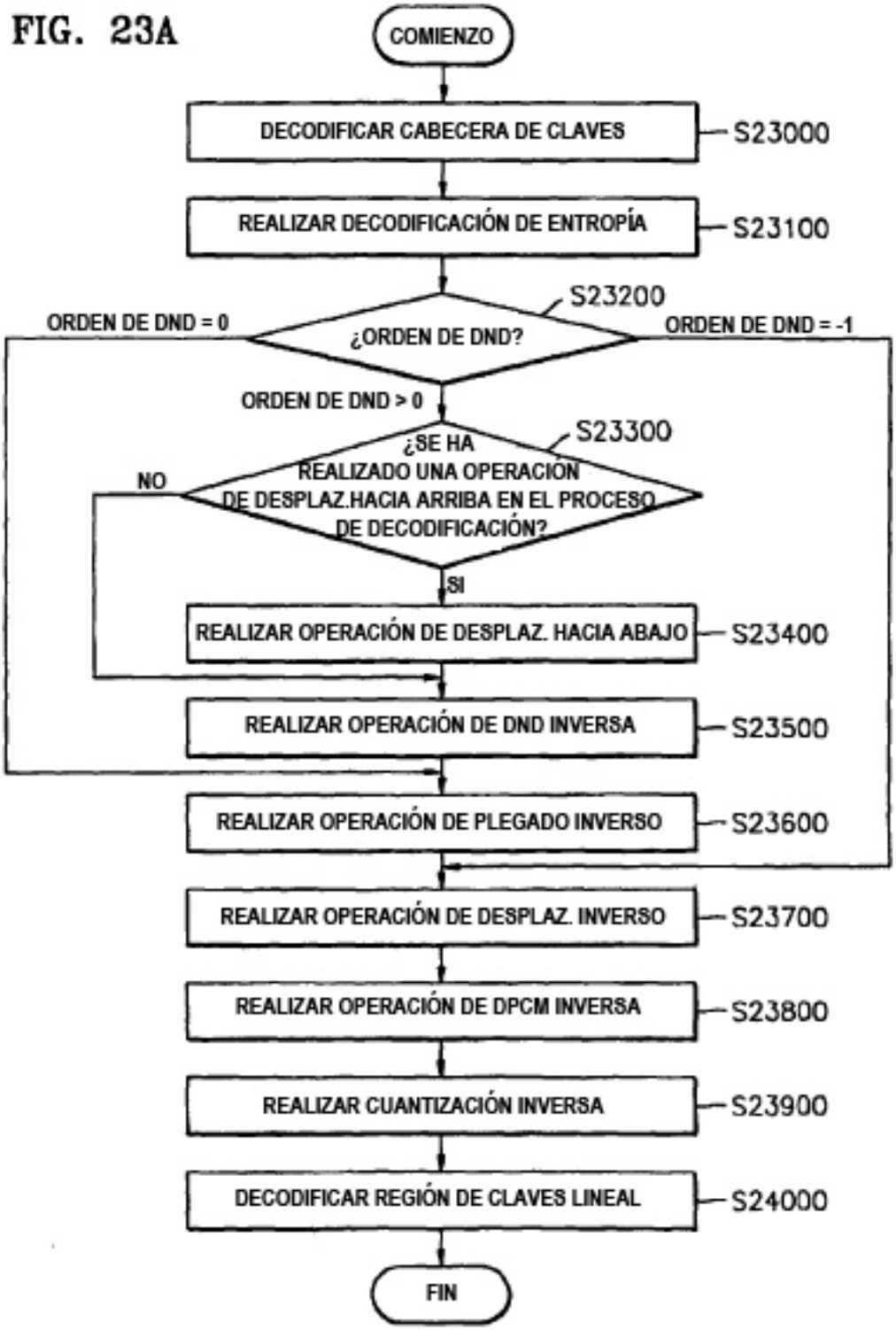
COMIENZO
DECODIFICAR CABECERA DE CLAVES
S23000
REALIZAR DECODIFICACIÓN DE ENTROPÍA
S23100
S23200
¿ORDEN DE DND?
ORDEN DE DND = 0
ORDEN DE DND = -1
ORDEN DE DND > 0
S23300
¿SE HA REALIZADO UNA OPERACIÓN DE DESPLAZ.HACIA ARRIBA EN EL PROCESO DE DECODIFICACIÓN?
NO
SI
REALIZAR OPERACIÓN DE DESPLAZ. HACIA ABAJO
S23400
REALIZAR OPERACIÓN DE DND INVERSA
S23500
REALIZAR OPERACIÓN DE PLEGADO INVERSO
S23600
REALIZAR OPERACIÓN DE DESPLAZ. INVERSO
S23700
REALIZAR OPERACIÓN DE DPCM INVERSA
S23800
REALIZAR CUANTIZACIÓN INVERSA
S23900
DECODIFICAR REGIÓN DE CLAVES LINEAL
S24000
FIN

FIG. 23B

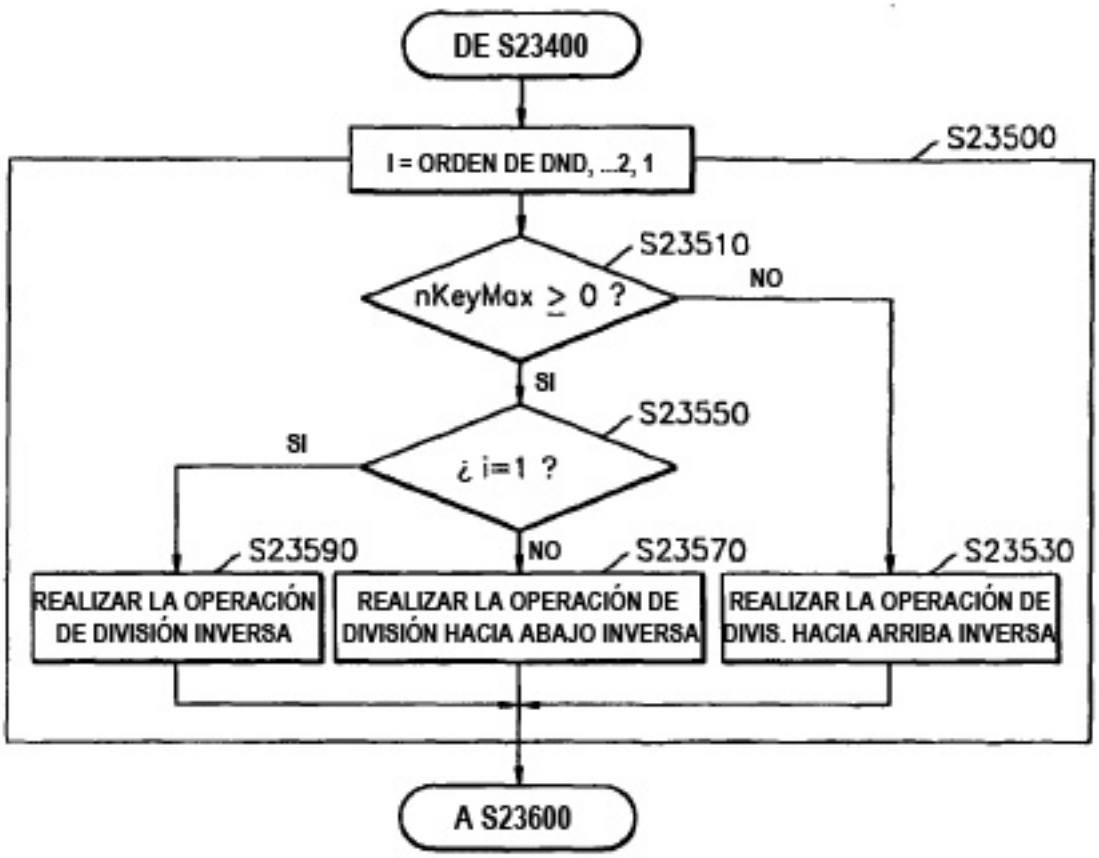
DE S23400
I = ORDEN DE DND, ...2, 1
S23500
S23510
nKeyMax ≥ 0 ?
NO
SI
S23550
SI
¿ i=1 ?
S23590
NO
S23570
S23530
REALIZAR LA OPERACIÓN DE DIVISIÓN INVERSA
REALIZAR LA OPERACIÓN DE DIVISIÓN HACIA ABAJO INVERSA
REALIZAR LA OPERACIÓN DE DIVIS. HACIA ARRIBA INVERSA
A S23600

FIG. 24A

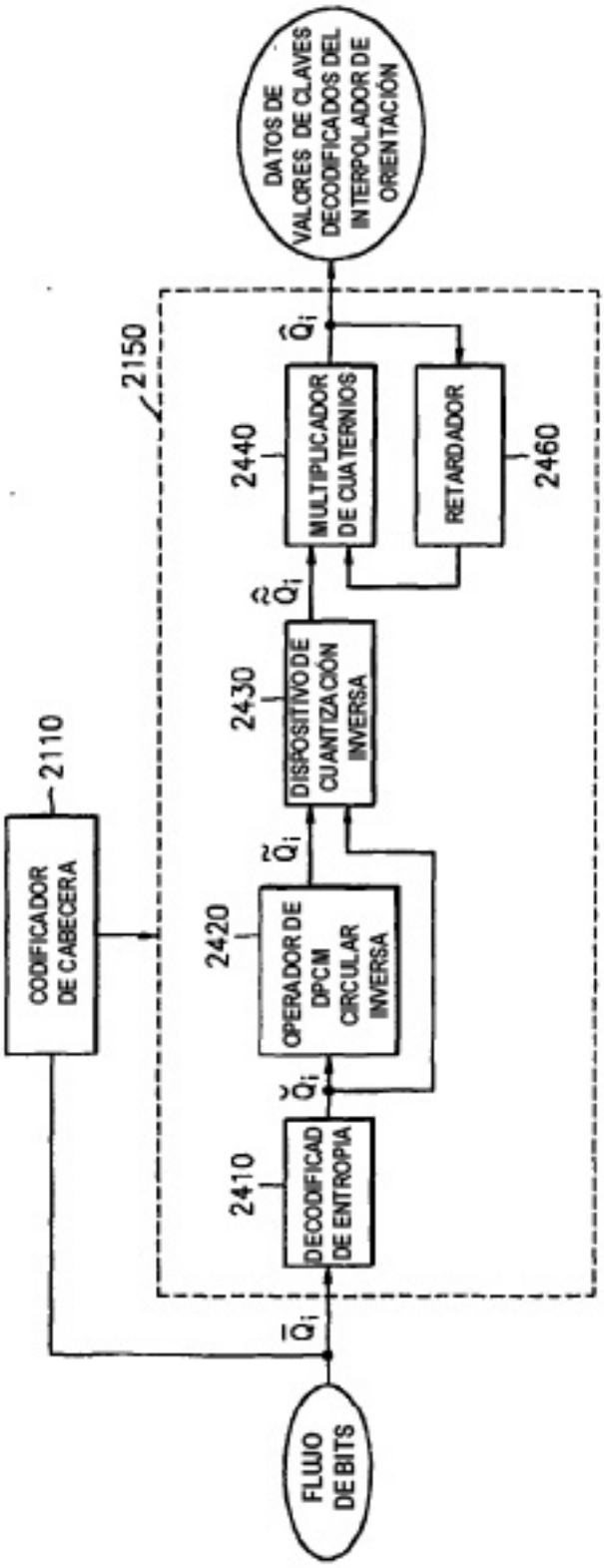
DATOS DE VALORES DE CLAVES DECODIFICADOS DEL INTERPOLADOR DE ORIENTACIÓN
2150
$\hat{Q}_i$
2440
MULTIPLICADOR DE CUATERNIOS
RETARDADOR
2460
$\tilde{\tilde{Q}}_i$
2430
DISPOSITIVO DE CUANTIZACIÓN INVERSA
2110
CODIFICADOR DE CABECERA
$\tilde{Q}_i$
2420
OPERADOR DE DPCM CIRCULAR INVERSA
$\check{Q}_i$
2410
DECODIFICAD DE ENTROPÍA
$\bar{Q}_i$
FLUJO DE BITS

FIG. 24B

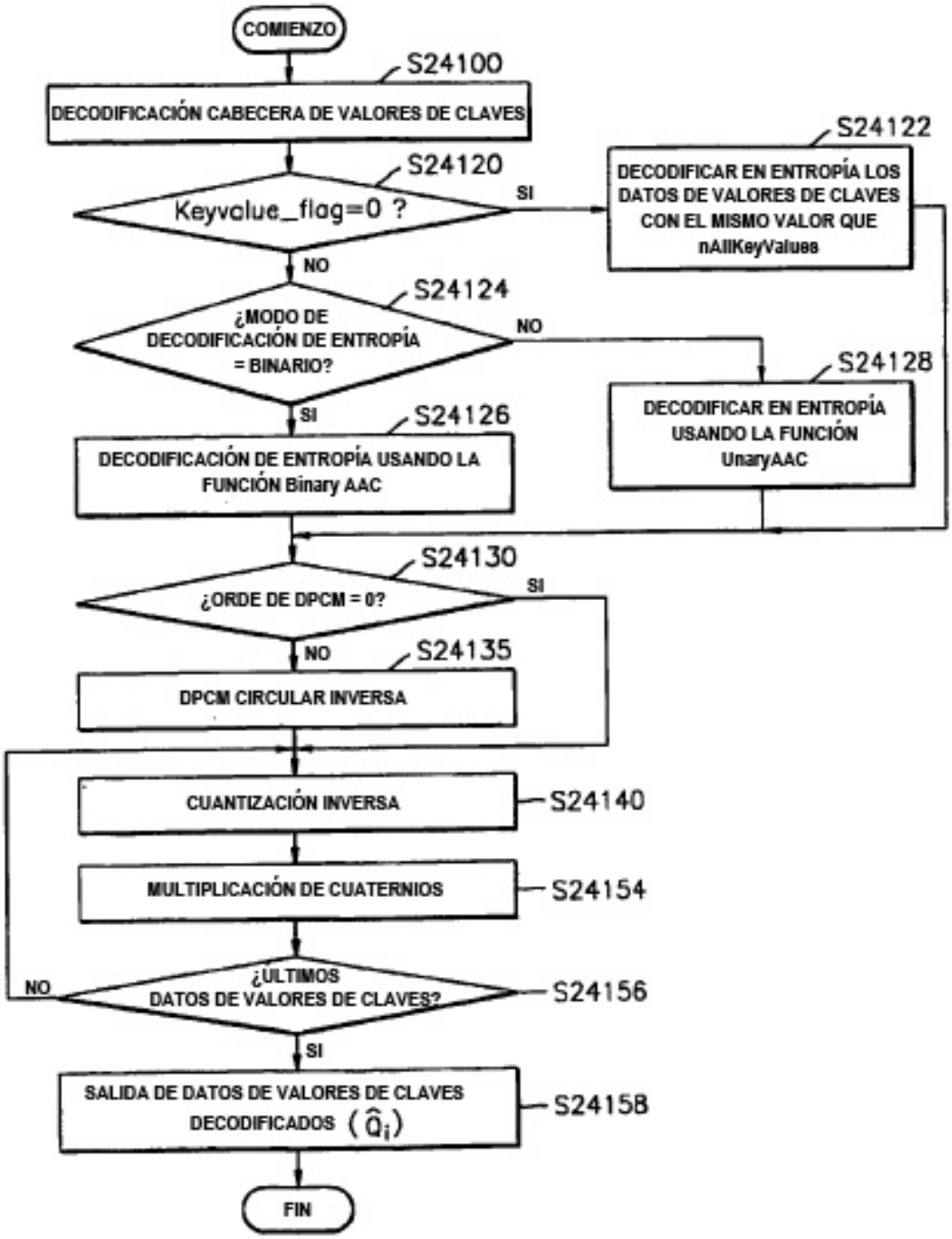
COMIENZO
S24100
DECODIFICACIÓN CABECERA DE VALORES DE CLAVES
S24120
Keyvalue_flag=0 ?
SI
S24122
DECODIFICAR EN ENTROPÍA LOS DATOS DE VALORES DE CLAVES CON EL MISMO VALOR QUE nAllKeyValues
NO
S24124
¿MODO DE DECODIFICACIÓN DE ENTROPÍA = BINARIO?
NO
S24128
DECODIFICAR EN ENTROPÍA USANDO LA FUNCIÓN UnaryAAC
SI
S24126
DECODIFICACIÓN DE ENTROPÍA USANDO LA FUNCIÓN Binary AAC
S24130
¿ORDE DE DPCM = 0?
SI
NO
S24135
DPCM CIRCULAR INVERSA
CUANTIZACIÓN INVERSA
S24140
MULTIPLICACIÓN DE CUATERNIOS
S24154
¿ÚLTIMOS DATOS DE VALORES DE CLAVES?
NO
S24156
SI
SALIDA DE DATOS DE VALORES DE CLAVES DECODIFICADOS $(\hat{Q}_i)$
S24158
FIN

FIG. 25

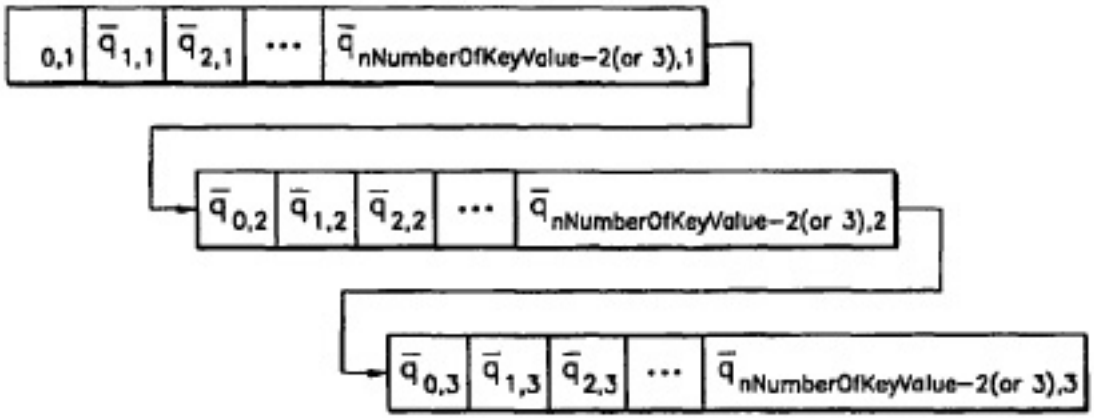
0,1
$\bar{q}_{1,1}$
$\bar{q}_{2,1}$
...
$\bar{q}_{nNumberOfKeyValue-2(or\ 3),1}$
$\bar{q}_{0,2}$
$\bar{q}_{1,2}$
$\bar{q}_{2,2}$
...
$\bar{q}_{nNumberOfKeyValue-2(or\ 3),2}$
$\bar{q}_{0,3}$
$\bar{q}_{1,3}$
$\bar{q}_{2,3}$
...
$\bar{q}_{nNumberOfKeyValue-2(or\ 3),3}$

FIG. 26

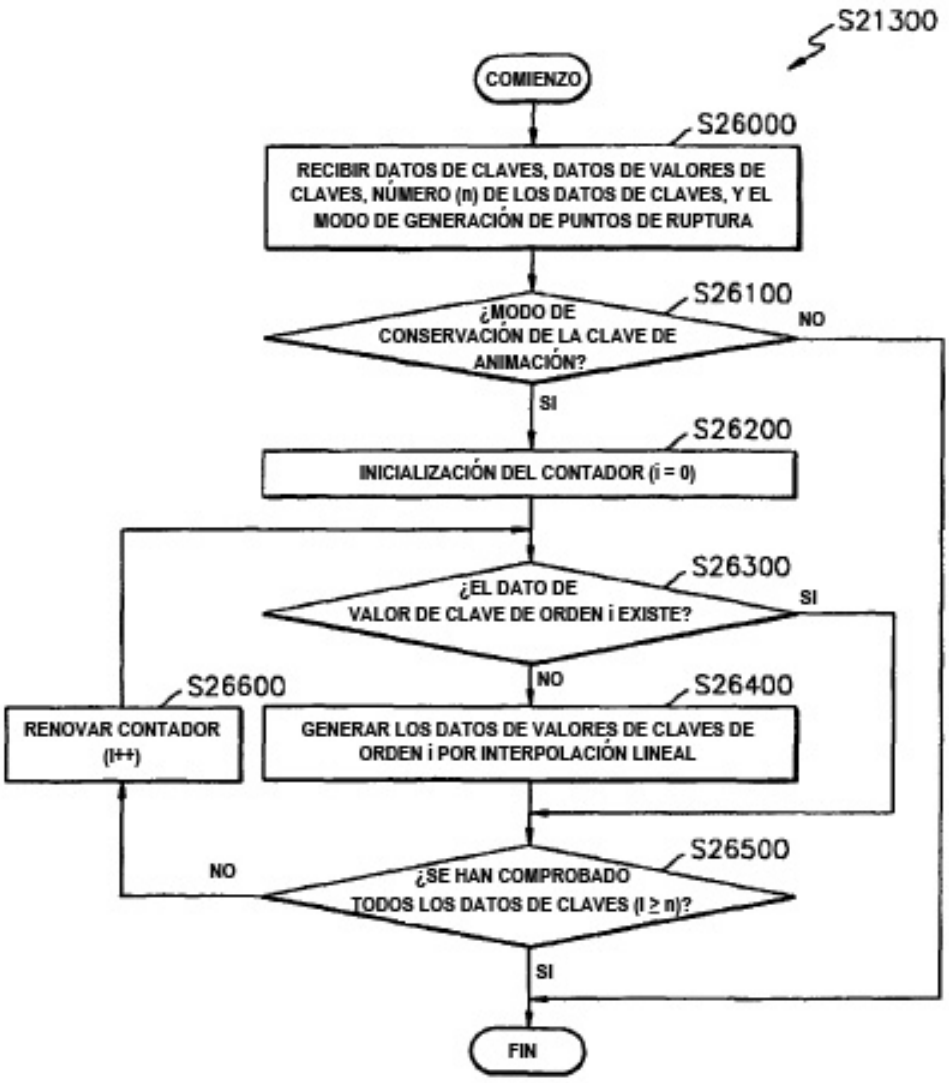
S21300
COMIENZO
S26000
RECIBIR DATOS DE CLAVES, DATOS DE VALORES DE CLAVES, NÚMERO (n) DE LOS DATOS DE CLAVES, Y EL MODO DE GENERACIÓN DE PUNTOS DE RUPTURA
S26100
¿MODO DE CONSERVACIÓN DE LA CLAVE DE ANIMACIÓN?
NO
SI
S26200
INICIALIZACIÓN DEL CONTADOR (i = 0)
S26300
¿EL DATO DE VALOR DE CLAVE DE ORDEN i EXISTE?
SI
NO
S26600
RENOVAR CONTADOR (i++)
S26400
GENERAR LOS DATOS DE VALORES DE CLAVES DE ORDEN i POR INTERPOLACIÓN LINEAL
S26500
¿SE HAN COMPROBADO TODOS LOS DATOS DE CLAVES (i ≥ n)?
NO
SI
FIN

FIG. 27

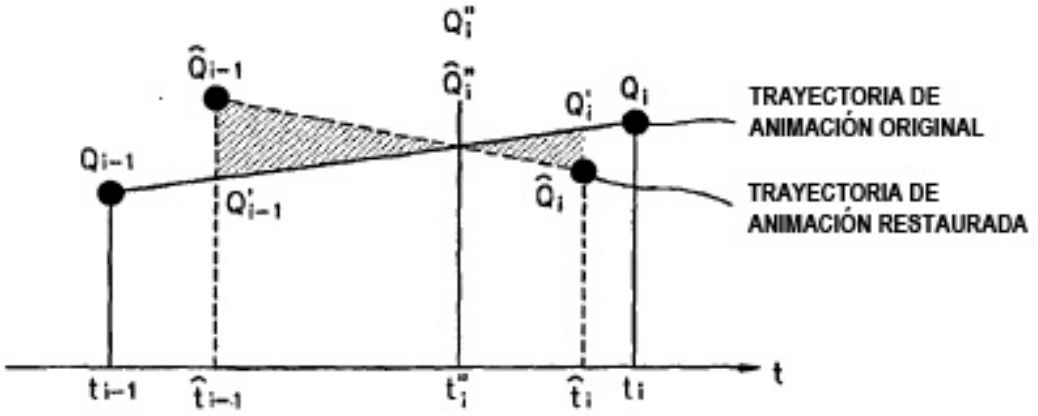
TRAYECTORIA DE ANIMACIÓN ORIGINAL
TRAYECTORIA DE ANIMACIÓN RESTAURADA
t

FIG. 28

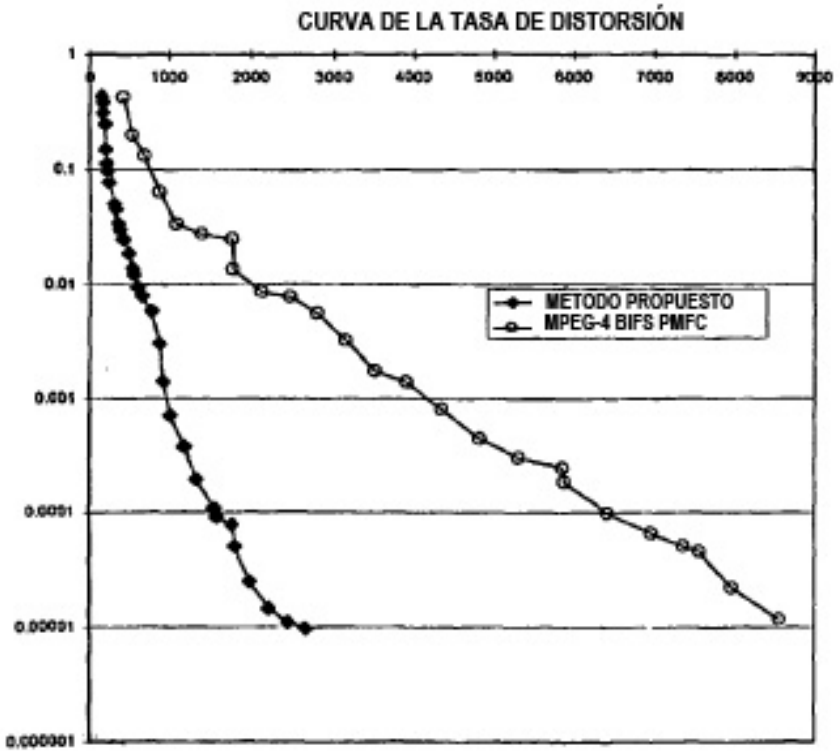
CURVA DE LA TASA DE DISTORSIÓN
MÉTODO PROPUESTO
MPEG-4 BIFS PMFC

FIG. 29A

```
clase CompressedOrientationInterpolator {
        KeyHeader kHeader;
        OriIKeyValueHeader oriIKVHeader;
        qf_start ();
        aligned (8) KeySelectionFlag ksKlag (kHeader, oriIKVHeader. bPreserveKey);
        Key k(kHeader);
        si (oriIKVHeader. nKVDPCMOrder == 0) // DPCM de primer orden
              OriIDPCMKeyValue oriIDPCMkeyValue (oriIKVHeader, oriIDPCMKVHeader,
 ksFlag. nNumberOfKeyValue – 1)
        si no // DPCM de segundo orden
              OriIDPCMKeyValue oriIDPCMkeyValue (oriIKVHeader, oriIDPCMKVHeader,
 ksFlag. nNumberOfKeyValue – 2)
 }
```

FIG. 29B

```
class KeyHeader {
        entero i:
        entero (5) sin signo nKeyQBit;
        entero (5) sin signo nNumKeyCodigBit;
        entero sin signo (nNumKeyCodingBit) nNumberOfKey;
        entero (4) sin signo nKeyDigit;
        bit (1) bisLinearKeySubRegion;
        si (bisLinearKeySubRegion = 1)
                LinearKey Ikey (nKeyDigit);
        bit (1) bRangeFlag;
        si (bRangeFlag = 1)
                KeyMinMax KeyMinMax (nKeyDigit);
        entero (5) sin signo nBitSize;
        entero (2) sin signo nKDPCMOrder;
        para (I = 0; I < nKDPCMOrder + 1; i++) {
                bit (1) nQIntraKeySign [[i]];
                si (i = 0 && nQIntraKeySign [i] ==1)
                        continuar;
                entero sin signo (nBitSize) nQIntraKey [[i]];
        }
        bit (1) bShiftFlag;
        si (bShiftFlag = 1) {
                bit (1) nKeyShiftSign;
                entero sin signo (nBitSize) nKeyShift:
        }
        entero (3) sin signo nDNDOrder;
        si (nDNDOrder = 7) {
                bit (1) bNoDND;
                si (bNoDND = 1)
                        nDNDOrder = -1;
        }
        entero nMaxQBit = nBitSize;
        para (i = 0; i < nDNDOrder; i++) {
                bit (1) nKeyMaxSign [[i]];
                entero sin signo (nMaxQBit) nKeyMax [[i]];
                nMaxQBit = (entero) (log 10 (abs (nKeyMax [i])) / log 10 (2)) + 1;
                si (nMaxQbit + 1 < nBitSize)
                        nMaxQBit + = 1;
                si no
                        nMaxQBit = nBitSize
        }
        entero bSignedAACFlag;
        entero nKeyCodingBitQBit = (entero) (log 10(nKeyQBit)) / log 10 (2)) + 1;
        entero sin signo (nKeyCodingBitQBit) nKeyCodigBit;
        si (nDNDOrder != -1 && nDNDOrder != 0) {
                bit (1) bkeyInvertDownFlag;
                si (bKeyInvertDownFlag == 1) {
                        entero sin signo (nKeyCodingBit) nKeyInvertDown;
                        bSignedAACFlag = 0;
                } si no {
                        bSignedAACFlag = 1;
                }
        } si no {
                bSignedAACFlag = 0;
        }
}
```

FIG. 29C

```
class LinearKey (entero nkeyDigit) {
        entero (5) sin signo nNumLinearKeyCodingBit;
        entero sin signo (nNumLinearKeyCodingBit) nNumberOfLinearkey;
        KeyMinMax kMinMax (nKeyDigit);
}
```

FIG. 29D

```
class KeyMinMax (entero nKeyDigit) {
        bit (1) bMinKeyDigitSame;
        si ((bMinKeyDigitSame = 0)
                entero (4) sin signo nMinKeyDigit;
        si no
                nMinKeyDigit = nKeyDigit;
        si (nMinKeyDigit != 0) {
                si (nMinKeyDigit < 8) {
                        entero count = (entero) (log 10 (10^nMinKeyDigit – 1) / log 10(2)) + 1;
                        bit (1) nMinKeyMantissaSign;
                        entero sin signo (count) nMinKeyMantissa;
                        bit (1) nMinKeyExponentSign;
                        entero (6) sin signo nMinKeyExponent;
                } si no
                        flotante (32) fKeyMin;
        }
        bit (1) bMaxKeyDigitSame;
        si (bMaxKeyDigitSame == 0)
                entero (4) sin signo nMaxKeyDigit;
        si no
                nMaxKeyDigit = nKeyDigit;
        si (nMaxKeyDigit != 0) {
                si (nMaxKeyDigit < 8) {
                        entero count = (entero) (log 10 (10^nMaxKeyDigit) – 1) / log 10 (2)) + 1
                        bit (1) nMaxKeyMantissaSign;
                        entero sin signo (count) nMaxKeyMantissa;
                        bit (1) bSameExponent;
                        si (bSameExponent == 0) {
                                bit (1) nMaxKeyExponentSign;
                                entero sin signo (6) nMaxKeyExponent;
                        }
                        si no
                                nMaxKeyExponent = nMinKeyExponent;
                } si no
                        flotante (32) fKeyMax;
        }
 }
```

FIG. 29E

```
Class OrilKeyValueHeader () {
bit (1) bPreserveKey;
entero (5) sin signo nKVQBit;
bit (1) nKVDPCMOrder;
OriIDPCMKeyValueHeader oriIDPCMKVHeader (nKVQBit, nKVDPCMOrder);
}
```

FIG. 29F

```
classe OrilDPCMKeyValueHeader (entero nKVQBit, entero sin signo nKVDPCMOrder) {
        entero sin signo (nKVQBit – 1) firstQKV_S;
        bit(1) nFirstXSign;
        entero sin signo (nKVQBit – 1) firstQKV_X;
        bit(1) nFirstYSign;
        entero sin signo (nKVQBit – 1) firstQKV_Y;
        bit(1) nFirstZSign;
        entero sin signo (nKVQBit – 1) firstQKV_Z;
        si (nKVDPCMOrder ==1) { // DPCM de segundo orden
        bit (1) nSecondXSign;
        entero sin signo (nKVQBit – 1) secondQKV_X;
        bit (1) nSecondYSign;
        entero sin signo (nKVQBit – 1) secondQKV_Y;
        bit (1) nSecondZSign;
        entero sin signo (nKVQBit – 1) secondQKV_Z;
        bit (1) bisMoreTwoKVs;
        }
        si (nKVDPCMOrder ==0 II bisMoreTwoKVs ==1) {
        bit (1) x_keyvalue_flag;
        OrilKeyValueCodingBit orilKVCodingBit_X (x_keyvlue_flag, nKVQBit);
        bit (1) y_keyvalue_flag;
        OrilKeyValueCodingBit orilKVCodingBit_Y (y_keyvlue_flag, nKVQBit);
        bit (1) z_keyvalue_flag;
        OrilKeyValueCodingBit orilKVCodingBit_Z (z_keyvlue_flag, nKVQBit);
        }
}
```

FIG. 29G

```
class OriIKeyValueCodingBit (entero sin signo flag_bit, entero nKVQBit) {
        entero count = (entero) (log10 (nKVQBit) / log10 (2)) + 1;
        si (flag_bit ==0) {
                entero sin signo (count) nKVCodingBit;
                si (nKVCodingBit ==1)
                        entero sin signo (nKVCodingBit) nAllKeyValue;
                si no {
                        bit (1) nSign;
                        entero sin signo (nKVCodingBit – 1) nAllKeyValue;
                }
        } si no {
                bit (1) bisUnaryAAC;
                si (bisUnaryAAC != 1)
                        entero sin signo (count) nKVCodingBit;
        }
}
```

FIG. 29H

```
Class KeySelectionFlag (KeyHeader kHeader, entero bPreserveKey) {
entero iI;
entero nNumOfKeyValue = 0;
si (bPreserveKey == 1) {
        para (I = 0, i < kHeader, nNumberOfKey; i++) {
                qf_decode (&keyFlag [i], keyFlagContext);
                si (keyFlag [i] ==1)
                        nNumOfKeyValue ++;
        }
} si no
        nNumOfKeyValue = kHeader. nNumberOfKey;
}
```

FIG. 29I

```
class Key (KeyHeader kHeader) {
        entero nQKey [kHeader. nNumberOfKey];
        entero i;
        entero nNumerOfRemainingKey;
        si (kHeader. bIsKinearKeySubRegion == 1)
                nNumberOfRemainingKey = kHeader. nNumberOfKey – kHeader. Ikey. nNumberOfLinearKey;
        si no
                nNumberOfRemainingKey = kHeader. nNumberOfKey;
        para (i = kHeader. nKDPCMOrder + 1; i < nNumberOfRemainingKey; i++) {
                si (kHeader. bSignedAACFlag ==0)
                        decodeUnsignedAAC (nQKey [i], kHeader. nKeyCodingBit, keyContext);
                si no
                        decodeSignedAAC (nQKey [i], kHeader. nKeyCodingBit + 1, keySignContext, keyContext);
        }
}
```

FIG. 29J

```
class OriIDPCMKeyValue (OrilDPCMKeyValueHeader kvHeader, entero nNumKV) {
        entero I;
        si (kvHeader, x_keyvalue_flag != 0) {
                si (kvHeader. orIKVCoding.Bit_X.bisUnaryAAC == 1)
                        para (i = 0, I < nNumKV; I++)
                                decodeUnariAAC (&DeltaKeyValue [i]. x, kVXSignConext, kVXUnaryContext);
                si no
                        para (i = 0, I < nNumKV; i++)
                                decodeSignedAAC (&DeltaKeyValue [i]. x, kvHeader. orIKVCodingBit_X. nKVCodingBit,
                                kVXSignContext, kVXContext);
        }
        si (kvHeader, y_keyvalue_flag != 0) {
                si (kvHeader. orIKVCoding.Bit_Y.bisUnaryAAC == 1)
                        para (i = 0, I < nNumKV; I++)
                                decodeUnariAAC (&DeltaKeyValue [i]. y, kVYSignConext, kVYUnaryContext);
                si no
                        para (i = 0, I < nNumKV; I++)
                                decodeSignedAAC (&DeltaKeyValue [i]. y, kvHeader. orIKVCodingBit_Y.nKVCodingBit,
                                kVYSignContext, kVYContext);
        }
        si (kvHeader, z_keyvalue_flag != 0) {
                si (kvHeader. orIKVCoding.Bit_Z.bisUnaryAAC == 1)
                        para (i = 0, I < nNumKV; I++)
                                decodeUnariAAC (&DeltaKeyValue [i]. x, kVZSignConext, kVZUnaryContext);
                si no
                        para (i = 0, I < nNumKV; I++)
                                decodeSignedAAC (&DeltaKeyValue [i]. z, kvHeader. orIKVCodingBit_Z. nKVCodingBit,
                                kVZSignContext, kVZContext);
        }
}
```