(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 794 559**

(51) Int. CI.:

H04N 19/13 (2014.01)
H04N 19/18 (2014.01)
H04N 19/593 (2014.01)
H04N 19/70 (2014.01)
H04N 19/196 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **01.09.2016 PCT/US2016/049962**

(87) Fecha y número de publicación internacional: **09.03.2017 WO17040828**

(96) Fecha de presentación y número de la solicitud europea: **01.09.2016 E 16766181 (8)**

(97) Fecha y número de publicación de la concesión europea: **26.02.2020 EP 3345394**

(54) Título: **Codificación de nivel de coeficiente en codificación de vídeo**

(30) Prioridad:

01.09.2015 US 201562212996 P
31.08.2016 US 201615252986

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.11.2020

(73) Titular/es:

QUALCOMM INCORPORATED (100.0%)
5775 Morehouse Drive
San Diego, CA 92121-1714, US

(72) Inventor/es:

ZHANG, LI;
KARCZEWICZ, MARTA y
CHEN, JIANLE

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Codificación de nivel de coeficiente en codificación de vídeo

[0001] Esta solicitud reivindica el beneficio de la solicitud provisional de patente estadounidense 62/212.996, presentada el 1 de septiembre de 2015.

CAMPO TÉCNICO

[0002] Esta divulgación se refiere a la codificación y a la descodificación de vídeo.

ANTECEDENTES

[0003] Las capacidades de vídeo digital se pueden incorporar a una amplia gama de dispositivos, que incluye televisores digitales, sistemas de radiodifusión digital directa, sistemas de radiodifusión inalámbrica, asistentes digitales personales (PDA), ordenadores portátiles o de escritorio, tabletas electrónicas, lectores de libros electrónicos, cámaras digitales, dispositivos de grabación digital, reproductores de medios digitales, dispositivos de videojuegos, consolas de videojuegos, teléfonos celulares o de radio por satélite, los denominados "teléfonos inteligentes", dispositivos de videoconferencia, dispositivos de transmisión continua de vídeo y similares. Los dispositivos de vídeo digital implementan técnicas de compresión de vídeo, tales como las descritas en las normas definidas por MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Parte 10, Codificación Avanzada de Vídeo (AVC), la norma de Codificación de Vídeo de Alta Eficacia (HEVC) y las ampliaciones de dichas normas. Los dispositivos de vídeo pueden transmitir, recibir, codificar, descodificar y/o almacenar información de vídeo digital con más eficacia implementando dichas técnicas de compresión de vídeo.

[0004] Las técnicas de compresión de vídeo realizan predicción espacial (intraimagen) y/o predicción temporal (interimagen) para reducir o eliminar la redundancia intrínseca a las secuencias de vídeo. Para la codificación de vídeo basada en bloques, un fragmento de vídeo (es decir, una trama de vídeo o una porción de una trama de vídeo) se puede dividir en bloques de vídeo. Los bloques de vídeo en un fragmento intracodificado (I) de una imagen se codifican usando predicción espacial con respecto a muestras de referencia en bloques vecinos en la misma imagen. Los bloques de vídeo en un fragmento intercodificado (P o B) de una imagen pueden usar predicción espacial con respecto a muestras de referencia de bloques vecinos en la misma imagen o predicción temporal con respecto a muestras de referencia en otras imágenes de referencia. Las imágenes se pueden denominar "tramas".

[0005] La predicción espacial o temporal da como resultado un bloque predictivo para un bloque a codificar. Los datos residuales representan diferencias de píxeles entre el bloque original a codificar y el bloque predictivo. Un bloque intercodificado se codifica de acuerdo con un vector de movimiento que apunta a un bloque de muestras de referencia que forman el bloque predictivo, y los datos residuales indican la diferencia entre el bloque codificado y el bloque predictivo. Un bloque intracodificado se codifica de acuerdo con un modo de intracodificación y de acuerdo con datos residuales. Para una mayor compresión, los datos residuales se pueden transformar desde el dominio del píxel hasta un dominio de transformada, dando como resultado coeficientes residuales, que a continuación se pueden cuantificar.

[0006] El documento de Flynn D. *et AL*.: "High Efficiency Video Coding (HEVC) Range Extensions text specification: Draft 7" (10-04-2014), XP030116230.17. JCT-VC MEETING; 27-3-2014 - 4-4-2014; VALENCIA; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/ SC29/WG11 AND ITU-T SG.16) [Reunión JCT-VC; 27-3-2014 - 4-4-2014; Valencia; (Equipo colaborativo conjunto en codificación de vídeo de ISO/IEC JTC1/SC29/WG11 e ITU-T SG.16], forma parte del proceso de estandarización de HEVC y muestra el proceso de binarización para el elemento sintáctico *coeff_abs_level_remaining*, en el que se usa una inicialización basada en estadísticas del parámetro de Rice y una actualización basada en plantillas (se considera el coeficiente anterior).

BREVE EXPLICACIÓN

[0007] Esta divulgación describe técnicas relacionadas con la codificación y descodificación por entropía en la codificación de vídeo híbrida basada en bloques. Por ejemplo, esta divulgación describe técnicas de binarización de elementos sintácticos para niveles de coeficientes en procesos de codificación y descodificación por entropía. Como parte de la binarización de un elemento sintáctico para un nivel de coeficiente, un codificador de vídeo (es decir, un codificador de vídeo o un descodificador de vídeo) puede determinar un parámetro K de Rice para el elemento sintáctico. Un código Rice de K-ésimo orden puede representar el elemento sintáctico. En algunos ejemplos, las técnicas propuestas de esta divulgación también pueden aplicarse en otros procedimientos de binarización en los que se usa el orden K. Como se describe en el presente documento, en algunos ejemplos, el codificador de vídeo puede usar un procedimiento de derivación de parámetro de Rice basado en plantillas y un procedimiento de derivación de parámetro de Rice basado en estadísticas para niveles de coeficientes de la misma unidad de transformada.

[0008] En un ejemplo, esta divulgación describe un procedimiento para descodificar datos de vídeo de acuerdo con las características definidas en la reivindicación 1.

[0009] En otro ejemplo, esta divulgación describe un procedimiento para codificar datos de vídeo de acuerdo con las características definidas en la reivindicación 6.

[0010] En otro ejemplo, esta divulgación describe un dispositivo para codificar datos de vídeo de acuerdo con las características definidas en la reivindicación 9.

[0011] Los detalles de uno o más ejemplos de la divulgación se exponen en los dibujos adjuntos y en la descripción siguiente. Otros rasgos característicos, objetivos y ventajas resultarán evidentes a partir de la descripción, los dibujos y las reivindicaciones.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0012]

La FIG. 1 es un diagrama de bloques que ilustra un sistema de codificación de vídeo ejemplar que puede usar las técnicas descritas en esta divulgación.

La FIG. 2 es un diagrama conceptual que ilustra un esquema de transformada basado en un árbol cuaternario residual, tal como el usado en la codificación de vídeo de alta eficacia (HEVC).

La FIG. 3 es un diagrama conceptual que ilustra un escaneo de coeficientes basado en grupos de codificación, tal como el usado en la HEVC.

La FIG. 4 es un diagrama conceptual que ilustra una plantilla local ejemplar.

La FIG. 5 es un diagrama de bloques que ilustra un codificador de vídeo ejemplar que puede implementar las técnicas descritas en la presente divulgación.

La FIG. 6 es un diagrama de bloques que ilustra un descodificador de vídeo ejemplar que puede implementar las técnicas descritas en la presente divulgación.

La FIG. 7 es un diagrama de flujo que ilustra una operación ejemplar para la codificación de datos de vídeo de acuerdo con una técnica de esta divulgación.

La FIG. 8 es un diagrama de flujo que ilustra una operación ejemplar de descodificación de datos de vídeo de acuerdo con una técnica de esta divulgación.

La FIG. 9 es un diagrama de flujo que ilustra una operación ejemplar de descodificación de datos de vídeo de acuerdo con una técnica de esta divulgación.

La FIG. 10 es un diagrama de flujo que ilustra una operación ejemplar para la codificación de datos de vídeo de acuerdo con una técnica de esta divulgación.

La FIG. 11 es un diagrama de flujo que ilustra una operación ejemplar en la que un codificador de vídeo usa un procedimiento de derivación basado en estadísticas de acuerdo con una técnica de esta divulgación.

La FIG. 12 es un diagrama de flujo que ilustra una operación ejemplar de un codificador de vídeo de acuerdo con un procedimiento de derivación basado en plantillas, de acuerdo con una técnica de esta divulgación.

La FIG. 13 es un diagrama de flujo que ilustra una operación ejemplar para determinar un parámetro de Rice basado en una función genérica, de acuerdo con una técnica de esta divulgación.

La FIG. 14 es un diagrama de flujo que ilustra una operación ejemplar para binarizar o desbinarizar una serie de elementos sintácticos, de acuerdo con una técnica de esta divulgación.

DESCRIPCIÓN DETALLADA

[0013] Esta divulgación describe técnicas que pueden relacionarse con un módulo de codificación por entropía en codificación de vídeo híbrida basada en bloques, especialmente para codificación de nivel de coeficiente. Las técnicas se pueden aplicar a los códecs de vídeo existentes, tales como los códecs de HEVC (codificación de vídeo de alta eficacia) o pueden ser una herramienta de codificación eficaz en futuros estándares de codificación de vídeo.

[0014] En diversas técnicas de codificación de vídeo, los elementos sintácticos particulares se codifican por entropía para reducir el número de bits necesarios para representar los elementos sintácticos. Como parte de la entropía que codifica un elemento sintáctico, un codificador de vídeo puede binarizar el elemento sintáctico. La binarización de un elemento sintáctico se refiere a un proceso de determinación de un código binario de longitud variable que representa

el elemento sintáctico. Por tanto, un elemento sintáctico binarizado es el código binario de longitud variable determinado que representa el elemento sintáctico. El elemento sintáctico binarizado puede referirse a una palabra de código. El codificador de vídeo puede codificar un elemento sintáctico binarizado usando una técnica de codificación aritmética. Por ejemplo, el codificador de vídeo puede codificar un elemento sintáctico binario usando codificación aritmética binaria adaptativa al contexto (CABAC). Por el contrario, un descodificador de vídeo puede usar la técnica de codificación aritmética para descodificar el elemento sintáctico binarizado. A continuación, el descodificador de vídeo puede desbinarizar el elemento sintáctico binarizado para recuperar el valor original del elemento sintáctico. A continuación, el descodificador de vídeo puede usar el elemento sintáctico recuperado como parte de un proceso para reconstruir una o más imágenes de datos de vídeo.

[0015] Un codificador de vídeo puede binarizar un elemento sintáctico de diversas maneras. Por ejemplo, el codificador de vídeo puede usar codificación de Rice o codificación (Exp)-Golomb exponencial para binarizar un elemento sintáctico. La codificación de Rice depende de un parámetro de Rice. Los parámetros de Rice mayores son más adecuados para binarizar elementos sintácticos con valores mayores, mientras que los parámetros de Rice menores son más adecuados para binarizar elementos sintácticos con valores menores.

[0016] En la HEVC y en particular otros sistemas de codificación de vídeo basados en bloques, una imagen se codifica usando un conjunto de unidades de codificación (CU). Cada CU de una imagen puede corresponder a uno o más bloques de codificación de localización conjunta dentro de la imagen. Para codificar una CU, un codificador de vídeo puede determinar bloques predictivos para una o más unidades de predicción (PU) de la CU y puede determinar datos residuales para la CU basados en muestras en los bloques predictivos para las PU de la CU y muestras correspondientes en los bloques de codificación de la CU. Por ejemplo, cada muestra de los datos residuales de la CU puede ser igual a una diferencia entre una muestra en un bloque predictivo de una PU de la CU y una muestra correspondiente del bloque de codificación de la CU. Los datos residuales para la CU se pueden dividir en uno o más bloques de transformada, cada uno de los cuales corresponde a una unidad de transformada (TU) de la CU. A continuación, el codificador de vídeo puede aplicar una transformada, tal como una transformada de coseno discreto, a un bloque de transformada de una TU para determinar un bloque de coeficiente. Los valores de coeficientes en el bloque de coeficientes pueden denominarse "niveles de coeficientes". Por ejemplo, en HEVC, un coeficiente de transformada es una cantidad escalar, considerada como en un dominio de frecuencia, que está asociada con un índice de frecuencia unidimensional o bidimensional particular en una parte de transformada inversa del proceso de descodificación; un nivel de coeficiente de transformada puede ser una cantidad de número entero que representa un valor asociado con un índice de frecuencia bidimensional particular en el proceso de descodificación antes del ajuste a escala para el cálculo de un valor de coeficiente de transformada. En algunos ejemplos, el codificador de vídeo puede cuantificar los niveles de coeficientes en el bloque de coeficientes.

[0017] Además, el codificador de vídeo puede subdividir cada bloque de coeficientes en uno o más grupos de coeficientes (CG). En HEVC, cada CG es un subbloque de 4x4 de un bloque de coeficientes. Para cada CG respectivo de un bloque de coeficientes, el codificador de vídeo puede generar un elemento sintáctico que indica si el CG respectivo incluye uno o más niveles de coeficientes distintos de cero. Para cada CG respectivo que incluye uno o más niveles de coeficientes distintos de cero, el codificador de vídeo puede representar cada nivel de coeficiente respectivo del CG respectivo usando un conjunto respectivo de uno o más elementos sintácticos para el nivel de coeficiente respectivo. El conjunto de elementos sintácticos para un nivel de coeficiente puede incluir un elemento sintáctico que indica si el nivel de coeficiente no es cero (es decir, un elemento sintáctico indicador de importancia), un elemento sintáctico que indica si el nivel de coeficiente es mayor que 1 (es decir, un indicador mayor1), un elemento sintáctico que indica si el nivel de coeficiente es mayor que 2 (es decir, un indicador mayor2), un elemento sintáctico que indica un signo positivo o negativo del nivel de coeficiente (es decir, un indicador de signo) y un elemento sintáctico que indica un valor de resto para el nivel de coeficiente (es decir, un elemento sintáctico de resto). Si el indicador de importancia indica que el nivel de coeficiente es 0, el indicador mayor1, el indicador mayor2, el indicador de señal y el elemento sintáctico de resto pueden no estar presentes. Si el indicador de importancia indica que el nivel de coeficiente no es cero y el indicador mayor1 indica que el nivel de coeficiente no es mayor que 1, el indicador mayor2 no está presente y el elemento sintáctico de resto indica el nivel de coeficiente. Si el indicador de importancia indica que el nivel de coeficiente no es cero y el indicador mayor1 indica que el nivel de coeficiente es mayor que 1, el indicador mayor2 está presente. Si el indicador mayor2 está presente pero indica que el nivel de coeficiente no es mayor que 2, el elemento sintáctico de resto indica el nivel de coeficiente menos 1. Si el indicador mayor2 está presente e indica que el nivel de coeficiente es mayor que 2, el elemento sintáctico de resto indica el nivel de coeficiente menos 2.

[0018] En la HEVC, un codificador de vídeo puede binarizar elementos sintácticos de resto que tienen valores pequeños usando códigos Rice y puede binarizar elementos sintácticos de resto que tienen valores mayores usando códigos Exp-Golomb. Como se menciona anteriormente, el proceso de determinar un código Rice para un valor depende de un parámetro de Rice. En la HEVC, un codificador de vídeo usa el llamado "procedimiento de derivación basado en retrospectiva" para establecer un parámetro de Rice usado para binarizar un elemento sintáctico de resto. En las extensiones de rango de HEVC, un codificador de vídeo usa el llamado "procedimiento de derivación basado en estadísticas" para establecer un parámetro de Rice usado para binarizar un elemento sintáctico de resto. Otra técnica, denominada "procedimiento de derivación basado en plantillas" determina un parámetro de Rice para binarizar un elemento sintáctico de resto basado en valores absolutos de niveles de coeficientes vecinos cubiertos por una plantilla local. Una plantilla puede incluir varias posiciones relativas en comparación con la posición del coeficiente

actual en un bloque de transformada. En algunos ejemplos, una plantilla es un conjunto contiguo de muestras que ocurren antes, en orden de análisis o descodificación, a un nivel de coeficiente actual.

[0019] Como se describe con más detalle a continuación, se pueden mejorar las técnicas para derivar un parámetro de Rice en HEVC, extensiones de rango de HEVC y otras propuestas anteriores diversas. Las técnicas descritas en esta divulgación pueden representar una mejora sobre dichas técnicas. Por ejemplo, de acuerdo con un ejemplo de esta divulgación, un codificador de vídeo (por ejemplo, un codificador de vídeo o un descodificador de vídeo) puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficientes de una única TU de una CU actual de una imagen. En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas. Por tanto, en este ejemplo, el codificador de vídeo puede usar el procedimiento de derivación basado en estadísticas para algunos elementos sintácticos de resto en la TU y puede usar el procedimiento de derivación basado en plantillas para otros elementos sintácticos de resto en la misma TU. Esta técnica puede permitir que el codificador de vídeo seleccione los parámetros de Rice que pueden dar como resultado una mejor compresión.

[0020] La FIG. 1 es un diagrama de bloques que ilustra un sistema de codificación de vídeo ejemplar 10 que puede utilizar las técnicas de esta divulgación. Como se usa en el presente documento, el término "codificador de vídeo" se refiere genéricamente tanto a codificadores de vídeo como a descodificadores de vídeo. En esta divulgación, los términos "codificación de vídeo" o "codificación" se pueden referir genéricamente a codificación de vídeo o a descodificación de vídeo.

[0021] Como se muestra en la FIG. 1, el sistema de codificación de vídeo 10 incluye un dispositivo de origen 12 y un dispositivo de destino 14. El dispositivo de origen 12 genera datos de vídeo codificados. Por consiguiente, el dispositivo de origen 12 se puede denominar dispositivo de codificación de vídeo o aparato de codificación de vídeo. El dispositivo de destino 14 puede descodificar los datos de vídeo codificados generados por el dispositivo de origen 12. Por consiguiente, el dispositivo de destino 14 se puede denominar dispositivo de descodificación de vídeo o aparato de descodificación de vídeo. El dispositivo de origen 12 y el dispositivo de destino 14 pueden ser ejemplos de dispositivos de codificación de vídeo o aparatos de codificación de vídeo.

[0022] El dispositivo de origen 12 y el dispositivo de destino 14 pueden comprender una amplia gama de dispositivos, que incluye ordenadores de escritorio, dispositivos informáticos móviles, ordenadores tipo *notebook* (por ejemplo, portátiles), tabletas electrónicas, descodificadores multimedia, aparatos telefónicos tales como los denominados teléfonos "inteligentes", televisores, cámaras, dispositivos de visualización, reproductores de medios digitales, consolas de videojuegos, ordenadores para vehículos o similares.

[0023] El dispositivo de destino 14 puede recibir datos de vídeo codificados desde el dispositivo de origen 12 por medio de un canal 16. El canal 16 puede comprender uno o más medios o dispositivos capaces de transferir los datos de vídeo codificados desde el dispositivo de origen 12 hasta el dispositivo de destino 14. En un ejemplo, el canal 16 puede comprender uno o más medios de comunicación que permiten al dispositivo de origen 12 transmitir datos de vídeo codificados directamente al dispositivo de destino 14 en tiempo real. En este ejemplo, el dispositivo de origen 12 puede modular los datos de vídeo codificados de acuerdo con una norma de comunicación, tal como un protocolo de comunicación inalámbrica, y puede transmitir los datos de vídeo modulados al dispositivo de destino 14. Los uno o más medios de comunicación pueden incluir medios de comunicación inalámbrica y/o alámbrica, tales como un espectro de radiofrecuencia (RF) o una o más líneas físicas de transmisión. Los uno o más medios de comunicación pueden formar parte de una red basada en paquetes, tal como una red de área local, una red de área amplia o una red global (por ejemplo, Internet). Los uno o más medios de comunicación pueden incluir encaminadores, conmutadores, estaciones base u otros equipos que facilitan la comunicación desde el dispositivo de origen 12 hasta el dispositivo de destino 14.

[0024] En otro ejemplo, el canal 16 puede incluir un medio de almacenamiento que almacena datos de vídeo codificados generados por el dispositivo de origen 12. En este ejemplo, el dispositivo de destino 14 puede acceder al medio de almacenamiento, por ejemplo, por medio de acceso a disco o acceso a tarjeta. El medio de almacenamiento puede incluir una variedad de medios de almacenamiento de datos de acceso local, tales como discos Blu-ray, DVD, CD-ROM, memoria flash u otros medios de almacenamiento digital adecuados para almacenar datos de vídeo codificados.

[0025] En otro ejemplo, el canal 16 puede incluir un servidor de archivos u otro dispositivo de almacenamiento intermedio que almacena los datos de vídeo codificados generados por el dispositivo de origen 12. En este ejemplo, el dispositivo de destino 14 puede acceder a datos de vídeo codificados almacenados en el servidor de archivos o en otro dispositivo de almacenamiento intermedio por medio de transmisión continua o descarga. El servidor de archivos puede ser un tipo de servidor capaz de almacenar datos de vídeo codificados y transmitir los datos de vídeo codificados al dispositivo de destino 14. Ejemplos de servidores de archivos incluyen servidores web (por ejemplo, para un sitio web), servidores de protocolo de transferencia de archivos (FTP), dispositivos de almacenamiento conectado a la red (NAS) y unidades de disco local.

[0026] El dispositivo de destino 14 puede acceder a los datos de vídeo codificados a través de una conexión de datos estándar, tal como una conexión a Internet. Ejemplos de tipos de conexiones de datos pueden incluir canales inalámbricos (por ejemplo, conexiones Wi-Fi), conexiones alámbricas (por ejemplo, DSL, módem por cable, etc.) o combinaciones de ambos que sean adecuadas para acceder a datos de vídeo codificados almacenados en un servidor de archivos. La transmisión de datos de vídeo codificados desde el servidor de archivos puede ser una transmisión continua, una transmisión de descarga o una combinación de ambas.

[0027] Las técnicas de esta divulgación no están limitadas a aplicaciones o configuraciones inalámbricas. Las técnicas se pueden aplicar a la codificación de vídeo como soporte de una variedad de aplicaciones multimedia, tales como radiodifusiones de televisión por aire, transmisiones de televisión por cable, transmisiones de televisión por satélite, transmisiones de vídeo en continuo, por ejemplo, por medio de Internet, codificación de datos de vídeo para su almacenamiento en un medio de almacenamiento de datos, descodificación de datos de vídeo almacenados en un medio de almacenamiento de datos u otras aplicaciones. En algunos ejemplos, el sistema de codificación de vídeo 10 puede estar configurado para admitir transmisión de vídeo unidireccional o bidireccional para admitir aplicaciones tales como la transmisión continua de vídeo, la reproducción de vídeo, la radiodifusión de vídeo y/o la videotelefonía.

[0028] El sistema de codificación de vídeo 10 ilustrado en la FIG. 1 es simplemente un ejemplo y las técnicas de esta divulgación pueden aplicarse a configuraciones de codificación de vídeo (por ejemplo, codificación de vídeo o descodificación de vídeo) que no incluyan necesariamente ninguna comunicación de datos entre los dispositivos de codificación y descodificación. En otros ejemplos, los datos se recuperan de una memoria local, se emiten en continuo por una red, o similares. Un dispositivo de codificación de vídeo puede codificar y almacenar datos en una memoria, y/o un dispositivo de descodificación de vídeo puede recuperar y descodificar datos de una memoria. En muchos ejemplos, la codificación y la descodificación se realizan mediante dispositivos que no se comunican entre sí, sino que simplemente codifican datos en una memoria y/o recuperan y descodifican datos de una memoria.

[0029] En el ejemplo de la FIG. 1, el dispositivo de origen 12 incluye una fuente de vídeo 18, un codificador de vídeo 20 y una interfaz de salida 22. En algunos ejemplos, la interfaz de salida 22 puede incluir un modulador/desmodulador (módem) y/o un transmisor. La fuente de vídeo 18 puede incluir un dispositivo de captura de vídeo, por ejemplo, una videocámara, un archivo de vídeo que contenga datos de vídeo previamente capturados, una interfaz de alimentación de vídeo para recibir datos de vídeo desde un proveedor de contenido de vídeo y/o un sistema de gráficos por ordenador para generar datos de vídeo, o una combinación de dichas fuentes de datos de vídeo.

[0030] El codificador de vídeo 20 puede codificar datos de vídeo procedentes de la fuente de vídeo 18. En algunos ejemplos, el dispositivo de origen 12 transmite directamente los datos de vídeo codificados al dispositivo de destino 14 por medio de la interfaz de salida 22. En otros ejemplos, los datos de vídeo codificados también pueden almacenarse en un medio de almacenamiento o en un servidor de archivos para su acceso posterior mediante el dispositivo de destino 14 para su descodificación y/o su reproducción.

[0031] En el ejemplo de la FIG. 1, el dispositivo de destino 14 incluye una interfaz de entrada 28, un descodificador de vídeo 30 y un dispositivo de visualización 32. En algunos ejemplos, la interfaz de entrada 28 incluye un receptor y/o un módem. La interfaz de entrada 28 puede recibir datos de vídeo codificados a través del canal 16. El dispositivo de visualización 32 puede estar integrado en, o ser externo a, el dispositivo de destino 14. En general, el dispositivo de visualización 32 muestra datos de vídeo descodificados. El dispositivo de visualización 32 puede comprender una variedad de dispositivos de visualización, tales como una pantalla de cristal líquido (LCD), una pantalla de plasma, una pantalla de diodos orgánicos emisores de luz (OLED) u otro tipo de dispositivo de visualización.

[0032] Tanto el codificador de vídeo 20 como el descodificador de vídeo 30 pueden implementarse como cualquiera de una variedad de circuitería adecuada, tales como uno o más microprocesadores, procesadores de señales digitales (DSP), circuitos integrados específicos de la aplicación (ASIC), matrices de puertas programables *in situ* (FPGA), lógica discreta, hardware o cualquier combinación de los mismos. Si las técnicas se implementan parcialmente en software, un dispositivo puede almacenar instrucciones para el software en un medio de almacenamiento legible por ordenador no transitorio adecuado, y puede ejecutar las instrucciones en hardware usando uno o más procesadores para realizar las técnicas de esta divulgación. Cualquiera de lo anterior (incluyendo hardware, software, una combinación de hardware y software, etc.) puede considerarse como uno o más procesadores. Tanto el codificador de vídeo 20 como el descodificador de vídeo 30 se pueden incluir en uno o más codificadores o descodificadores, de los que cualquiera se puede integrar como parte de un codificador/descodificador (CÓDEC) combinado en un dispositivo respectivo.

[0033] Esta divulgación puede referirse, en general, a "señalizar" o "transmitir" determinada información a otro dispositivo (por ejemplo, al descodificador de vídeo 30). El término "señalizar" o "transmitir" puede referirse en general a la comunicación de elementos sintácticos y/u otros datos usados para descodificar los datos de vídeo comprimidos. Dicha comunicación se puede producir en tiempo real o casi real. De forma alternativa, dicha comunicación puede producirse durante un tramo de tiempo, tal como podría producirse cuando se almacenan elementos sintácticos en un medio de almacenamiento legible por ordenador en un flujo de bits codificado en el momento de la codificación, que pueden ser recuperados después por un dispositivo de descodificación en cualquier momento tras haber sido almacenados en este medio.

[0034] En algunos ejemplos, el codificador de vídeo 20 y el descodificador de vídeo 30 funcionan de acuerdo con una norma de compresión de vídeo, es decir, una norma de codificación de vídeo. Entre las normas de codificación de vídeo ejemplares se incluyen ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 o ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual e ITU-T H.264 (también conocida como ISO/IEC MPEG-4 AVC), incluidas sus extensiones de codificación de vídeo escalable (SVC) y de codificación de vídeo multivista (MVC). Además, recientemente se ha desarrollado un nuevo estándar de codificación de vídeo, a saber, la codificación de vídeo de alta eficacia (HEVC) o ITU-T H.265, que incluye sus extensiones de rango, extensión de múltiples vistas (MV-HEVC) y extensión escalable (SHVC) por el equipo de colaboración conjunta sobre codificación de vídeo (JCT-VC), así como el equipo de colaboración conjunta sobre el desarrollo de la extensión de codificación de vídeo 3D (JCT-3V) del grupo de expertos en codificación de vídeo (VCEG) de UIT-T y el grupo de expertos en imágenes en movimiento ISO/IEC (MPEG). Un borrador de memoria descriptiva de HEVC, y denominado HEVC WD o "especificación de HEVC" más adelante en el presente documento, está disponible en http://phenix.int-evry.frjct/doc_end_user/documents/14_Vienna/wg11/JCTVC-N1003-v1.zip. Otro borrador de memoria descriptiva de HEVC, es Bross y col., "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Last Call)", Equipo de Colaboración Conjunta en Codificación de Vídeo (JCT-VC) de ITU-T SG 16 WP 3 e ISO/IEC JTC 1/SC 29/WG 11, 12.ª reunión: Ginebra, Suiza, 14-23 de enero de 2013, documento JCTVC-L1003_v34, (más adelante en el presente documento, "JCTVC-L1003"), que puede estar disponible en http://phenix.it-sudparis.eu/jct/doc_end_user/documents/12_Geneva/wg11/JCTVC-L1003-v34.zip. Como ejemplos, los aspectos de diseño de HEVC se presentan a continuación, centrándose en la codificación del coeficiente de transformada, así como en la codificación aritmética binaria adaptativa al contexto (CABAC).

[0035] En la HEVC y otras normas de codificación de vídeo, una secuencia de vídeo incluye típicamente una serie de imágenes. Las imágenes también se pueden denominar "tramas". Una imagen puede incluir una o más matrices de muestra. Por ejemplo, una imagen puede incluir una matriz de muestras de luma y dos matrices de muestras de croma, indicadas como S_L, S_{Cb} y S_{Cr}, respectivamente. S_L es una matriz bidimensional (es decir, un bloque) de muestras de luma. SCb es una matriz bidimensional de muestras de croma Cb. S_{Cr} es una matriz bidimensional de muestras de croma Cr. En otros casos, una imagen puede ser monocromática y puede incluir solo una matriz de muestras de luma.

[0036] En la HEVC, la unidad de codificación más grande en un fragmento se denomina unidad de árbol de codificación (CTU). Una CTU contiene un árbol cuaternario cuyos nodos son unidades de codificación. El tamaño de una CTU puede variar desde 16x16 a 64x64 en el perfil principal de la HEVC (aunque, técnicamente, se puede prestar soporte a tamaños de CTU de 8x8). Para generar una representación codificada de una imagen (es decir, para codificar la imagen), el codificador de vídeo 20 puede generar un conjunto de unidades de árbol de codificación (CTU). Cada una de las respectivas CTU puede ser un bloque de árbol de codificación de muestras de luma, dos bloques de árbol de codificación correspondientes de muestras de croma y estructuras sintácticas usadas para codificar las muestras de los bloques de árbol de codificación. En imágenes monocromáticas o imágenes que tienen tres planos de color separados, una CTU puede comprender un solo bloque de árbol de codificación y unas estructuras sintácticas usadas para codificar las muestras del bloque de árbol de codificación. Un bloque de árbol de codificación puede ser un bloque de muestras de NxN. Una CTU también puede denominarse "bloque de árbol" o "unidad de codificación de máximo tamaño" (LCU). Un segmento puede incluir un número entero de CTU ordenadas consecutivamente por orden de escaneo, tal como un orden de escaneo por trama.

[0037] Para generar una CTU codificada (es decir, para codificar una CTU) en la HEVC, el codificador de vídeo 20 puede realizar de forma recursiva una división de árbol cuaternario en los bloques de árbol de codificación de una CTU para dividir los bloques de árbol de codificación en bloques de codificación; de ahí el nombre de "unidades de árbol de codificación". Un bloque de codificación es un bloque de muestras de NxN. Una CU puede ser un bloque de codificación de muestras de luma y dos bloques de codificación correspondientes de muestras de croma de una imagen que tiene una matriz de muestras de luma, una matriz de muestras de Cb y una matriz de muestras de Cr, y estructuras de sintaxis usadas para codificar las muestras de los bloques de codificación. En imágenes monocromáticas o imágenes que tienen tres planos de color separados, una CU puede comprender un único bloque de codificación y estructuras sintácticas usadas para codificar las muestras del bloque de codificación. Una CU puede ser del mismo tamaño que una CTU, aunque una CU puede ser tan pequeña como 8x8.

[0038] El codificador de vídeo 20 puede generar una representación codificada de una CU (es decir, codificar la CU). Como parte de la codificación de una CU, el codificador de vídeo 20 puede dividir un bloque de codificación de una CU en uno o más bloques de predicción. Un bloque de predicción puede ser un bloque rectangular (es decir, cuadrado o no cuadrado) de muestras en las que se aplica la misma predicción. Una unidad de predicción (PU) de una CU puede ser un bloque de predicción de muestras de luma, dos bloques de predicción correspondientes de muestras de croma de una imagen y estructuras de sintaxis usadas para predecir las muestras de bloques de predicción. El codificador de vídeo 20 puede generar bloques predictivos de luma, Cb y Cr para bloques de predicción de luma, Cb y Cr de cada PU de la CU. En imágenes monocromáticas o imágenes que tienen tres planos de color separados, una PU puede comprender un único bloque de predicción y estructuras sintácticas usadas para predecir el bloque de predicción.

[0039] El codificador de vídeo 20 puede usar intrapredicción o interpredicción para generar los bloques predictivos para una PU. Cada CU se codifica con uno de los modos de intrapredicción o interpredicción. Si el codificador de vídeo 20 usa intrapredicción para generar los bloques predictivos de una PU, el codificador de vídeo 20 puede generar los bloques predictivos de la PU basándose en muestras descodificadas de la imagen asociada a la PU. Si el codificador de vídeo 20 usa interpredicción para generar los bloques predictivos de una PU, el codificador de vídeo 20 puede generar los bloques predictivos de la PU basándose en muestras descodificadas de una o más imágenes distintas a la imagen asociada a la PU. Cuando una CU se intercodifica (es decir, cuando se usa la interpredicción para generar bloques predictivos para las PU de la CU), la CU se puede dividir en 2 o 4 PU o la CU puede convertirse en una sola PU cuando no se aplica otra división. Cuando dos PU están presentes en una CU, las dos PU pueden ser rectángulos de mitad de tamaño o de tamaño de dos rectángulos con un ¼ o ¾ del tamaño de la CU. Además, cuando la CU está intercodificada, está presente un conjunto de información de movimiento para cada PU. Además, cuando una CU está intercodificada, cada PU de la CU se codifica con un modo de interpredicción separado para obtener el conjunto de información de movimiento.

[0040] Después de que el codificador de vídeo 20 genera bloques predictivos (por ejemplo, bloques predictivos de luma, Cb y Cr) para una o más PU de una CU, el codificador de vídeo 20 puede generar uno o más bloques residuales de la CU. Cada muestra en un bloque residual de la CU indica una diferencia entre una muestra en un bloque predictivo para una PU de la CU y una muestra correspondiente en un bloque de codificación de la CU. Por ejemplo, el codificador de vídeo 20 puede generar un bloque residual de luma de la CU. Cada muestra en el bloque residual de la CU indica una diferencia entre una muestra de luma en un bloque de luma predictivo de una PU de la CU y una muestra correspondiente en el bloque de codificación de luma de la CU. Además, el codificador de vídeo 20 puede generar un bloque residual de Cb de la CU. Cada muestra en el bloque residual de Cb de la CU puede indicar una diferencia entre una muestra de Cb en un bloque predictivo de Cb de una PU de la CU y una muestra correspondiente en el bloque de codificación de Cb de la CU. El codificador de vídeo 20 también puede generar un bloque residual de Cr de la CU. Cada muestra en el bloque residual de Cr de la CU puede indicar una diferencia entre una muestra de Cr en un bloque de Cr predictivo para una PU de la CU y una muestra correspondiente en el bloque de codificación de Cr de la CU.

[0041] Además, el codificador de vídeo 20 puede descomponer cada bloque residual de una CU en uno o más bloques de transformada (por ejemplo, usando división por árbol cuaternario). Un bloque de transformada puede ser un bloque rectangular (cuadrado o no cuadrado) de muestras a las que se aplica la misma transformada. Una unidad de transformada (TU) de una CU puede ser un bloque de transformada de muestras de luma, dos bloques de transformada correspondientes de muestras de croma y estructuras de sintaxis usadas para transformar las muestras de bloques de transformada. Por tanto, cada TU de una CU puede estar asociada a un bloque de transformada de luma, un bloque de transformada de Cb y un bloque de transformada de Cr. El bloque de transformada de luma asociado con la TU puede ser un subbloque del bloque residual de luma de la CU. El bloque de transformada de Cb puede ser un subbloque del bloque residual de Cb de la CU. El bloque de transformada de Cr puede ser un subbloque del bloque residual de Cr de la CU. En imágenes monocromáticas o imágenes que tienen tres planos de color separados, una TU puede comprender un único bloque de transformada y estructuras sintácticas usadas para transformar las muestras del bloque de transformada. En algunos ejemplos, los bloques residuales en la misma CU para los componentes de luma y croma pueden dividirse de diferentes maneras. En algunos ejemplos, existe una restricción de que el tamaño de la CU debe ser igual al tamaño de la PU y al tamaño de la TU; es decir, una CU solo contiene una PU y una TU.

[0042] El codificador de vídeo 20 puede aplicar una o más transformadas a un bloque de transformada para una TU para generar un bloque de coeficientes para la TU. El bloque de coeficiente para la TU puede comprender niveles de coeficiente de la TU. Por ejemplo, el codificador de vídeo 20 puede aplicar una o más transformadas a un bloque de transformada de luma para una TU para generar un bloque de coeficientes de luma para la TU. El codificador de vídeo 20 puede aplicar una o más transformadas a un bloque de transformada de Cb de una TU para generar un bloque de coeficientes de Cb para la TU. El codificador de vídeo 20 puede aplicar una o más transformadas a un bloque de transformada de Cr de una TU para generar un bloque de coeficientes de Cr para la TU. Un bloque de coeficientes puede ser una matriz bidimensional de coeficientes de transformada. Un coeficiente de transformada puede ser una cantidad escalar. Como se indica previamente, el valor de un coeficiente de transformada puede denominarse nivel de coeficiente. En algunos ejemplos, el codificador de vídeo 20 omite la aplicación de la transformada. Cuando se omite la aplicación de la transformada para un bloque, el bloque (por ejemplo, TU) es un "bloque de omisión de transformada". En consecuencia, en dichos ejemplos, el codificador de vídeo 20 y el descodificador de vídeo 30 pueden tratar las muestras residuales en un bloque de transformada de la misma manera que los coeficientes de transformada.

[0043] Después de generar un bloque de coeficientes, el codificador de vídeo 20 puede cuantificar el bloque de coeficientes. La cuantificación se refiere, en general, a un proceso en el que coeficientes de transformada se cuantifican para reducir, posiblemente, la cantidad de datos usados para representar los coeficientes de transformada, proporcionando más compresión. En algunos ejemplos, el codificador de vídeo 20 omite la etapa de cuantificar el bloque de coeficientes. Después de que el codificador de vídeo 20 cuantifique un bloque de coeficientes, el codificador de vídeo 20 puede codificar por entropía elementos sintácticos que indican los coeficientes de transformada cuantificados. Por ejemplo, el codificador de vídeo 20 puede realizar una codificación aritmética binaria adaptativa al contexto (CABAC) en los elementos sintácticos que indican los coeficientes de transformada cuantificados. El

codificador de vídeo 20 puede facilitar los elementos sintácticos codificados por entropía en un flujo de bits.

[0044] El codificador de vídeo 20 puede facilitar un flujo de bits que puede incluir elementos sintácticos codificados con entropía y elementos sintácticos codificados sin entropía. El flujo de bits puede incluir una secuencia de bits que forma una representación de imágenes codificadas y datos asociados. Por tanto, el flujo de bits puede formar una representación codificada de datos de vídeo. El flujo de bits puede comprender una secuencia de unidades de capa de abstracción de red (NAL). Cada una de las unidades NAL incluye una cabecera de unidad NAL y encapsula una carga útil de secuencia de octetos sin procesar (RBSP). La cabecera de unidad de NAL puede incluir un elemento sintáctico que indica un código de tipo de unidad de NAL. El código de tipo de unidad de NAL especificado por la cabecera de unidad de NAL de una unidad de NAL indica el tipo de la unidad de NAL. Una RBSP puede ser una estructura sintáctica que contiene un número entero de octetos que están encapsulados dentro de una unidad de NAL. En algunos casos, una RBSP incluye bits cero.

[0045] Diferentes tipos de unidades NAL pueden encapsular diferentes tipos de RBSP. Por ejemplo, un primer tipo de unidad de NAL puede encapsular una RBSP para un conjunto de parámetros de imagen (PPS), un segundo tipo de unidad de NAL puede encapsular una RBSP para un fragmento codificado, un tercer tipo de unidad de NAL puede encapsular una RBSP para información de mejora complementaria (SEI), y así sucesivamente. Las unidades de NAL que encapsulan las RBSP para datos de codificación de vídeo (a diferencia de las RBSP para conjuntos de parámetros y mensajes SEI) se pueden denominar unidades de NAL de capa de codificación de vídeo (VCL).

[0046] En el ejemplo de la FIG. 1, el descodificador de vídeo 30 recibe un flujo de bits generado por el codificador de vídeo 20. Además, el descodificador de vídeo 30 puede analizar sintácticamente el flujo de bits para obtener elementos sintácticos a partir del flujo de bits. Como parte de la obtención de los elementos sintácticos del flujo de bits, el descodificador de vídeo 30 puede realizar la decodificación de entropía (por ejemplo, descodificación CABAC) para obtener elementos sintácticos particulares del flujo de bits. El descodificador de vídeo 30 puede reconstruir (es decir, descodificar) las imágenes de los datos de vídeo en base a, al menos en parte, los elementos sintácticos obtenidos a partir del flujo de bits. El proceso para reconstruir los datos de vídeo puede ser, en general, recíproco al proceso realizado por el codificador de vídeo 20 para codificar los datos de vídeo. Por ejemplo, el descodificador de vídeo 30 puede usar intrapredicción o interpredicción para determinar bloques predictivos de las PU de una CU actual. Además, el descodificador de vídeo 30 puede cuantificar de forma inversa bloques de coeficientes para TU de la CU actual. El descodificador de vídeo 30 puede realizar transformadas inversas en los bloques de coeficientes para reconstruir los bloques de transformada para las TU de la CU actual. El descodificador de vídeo 30 puede reconstruir los bloques de codificación de la CU actual añadiendo las muestras de los bloques predictivos para PU de la CU actual a muestras correspondientes de los bloques de transformada para las TU de la CU actual. Reconstruyendo los bloques de codificación para cada CU de una imagen, el descodificador de vídeo 30 puede reconstruir la imagen.

[0047] Como se indica anteriormente, el codificador de vídeo 20 y el descodificador de vídeo 30 pueden realizar la codificación CABAC. CABAC es un procedimiento de codificación por entropía presentado por primera vez en H.264/AVC y usado ahora en HEVC. CABAC implica tres funciones principales: binarización, modelado de contexto y codificación aritmética. La binarización asigna elementos sintácticos a símbolos binarios (bins) que se denominan cadenas bin. En otras palabras, para aplicar la codificación CABAC a un elemento sintáctico, el codificador de vídeo puede binarizar el elemento sintáctico para formar una serie de uno o más bits, que se denominan "bins". El modelado de contexto estima la probabilidad de los bins. Finalmente, el codificador aritmético binario comprime los bins en bits en base a la probabilidad estimada.

[0048] Como parte del modelado de contexto, el codificador de vídeo puede identificar un contexto de codificación. El contexto de codificación puede identificar probabilidades de codificación de bins que tengan valores particulares. Por ejemplo, un contexto de codificación puede indicar una probabilidad de 0,7 de codificar un bin de valor 0 y una probabilidad de 0,3 de codificar un bin de valor 1. Después de identificar el contexto de codificación, el codificador de vídeo 20 puede dividir un intervalo en un subintervalo inferior y en un subintervalo superior. Uno de los subintervalos puede estar asociado con el valor 0 y el otro subintervalo puede estar asociado con el valor 1. El ancho de los subintervalos puede ser proporcional a las probabilidades indicadas para los valores asociados por el contexto de codificación identificado. Si un bin del elemento sintáctico tiene el valor asociado con el subintervalo inferior, el valor codificado puede ser igual al límite inferior del subintervalo inferior. Si el mismo bin del elemento sintáctico tiene el valor asociado con el subintervalo superior, el valor codificado puede ser igual al límite inferior del subintervalo superior. Para codificar el siguiente bin del elemento sintáctico, el codificador de vídeo 20 puede repetir estas etapas con el intervalo estando el subintervalo asociado con el valor del bit codificado. Cuando el codificador de vídeo repite estas etapas para el siguiente bin, el codificador de vídeo puede usar probabilidades modificadas basadas en las probabilidades indicadas por el contexto de codificación identificado y en los valores reales de los bins codificados.

[0049] Cuando un descodificador de vídeo realiza la descodificación CABAC en un elemento sintáctico, el descodificador de vídeo 30 puede identificar un contexto de codificación. El descodificador de vídeo puede a continuación dividir un intervalo en un subintervalo inferior y un subintervalo superior. Uno de los subintervalos puede estar asociado con el valor 0 y el otro subintervalo puede estar asociado con el valor 1. El ancho de los subintervalos puede ser proporcional a las probabilidades indicadas para los valores asociados por el contexto de codificación identificado. Si el valor codificado está dentro del subintervalo más bajo, el descodificador de vídeo 30 puede

descodificar un bin que tenga el valor asociado con el subintervalo inferior. Si el valor codificado está dentro del subintervalo superior, el descodificador de vídeo 30 puede descodificar un bin que tenga el valor asociado con el subintervalo superior. Para descodificar un siguiente bin del elemento sintáctico, el descodificador de vídeo 30 puede repetir estas etapas siendo el intervalo el subintervalo que contenga el valor codificado. Cuando el descodificador de vídeo 30 repita estas etapas para el siguiente bin, el descodificador de vídeo 30 puede usar probabilidades modificadas basadas en las probabilidades indicadas por el contexto de codificación identificado y en los bins descodificados. El descodificador de vídeo 30 puede desbinarizar a continuación los bins para recuperar el elemento sintáctico.

[0050] En lugar de llevar a cabo una codificación CABAC convencional en todos los elementos sintácticos, el codificador de vídeo 20 puede codificar algunos bins usando codificación CABAC de derivación. Puede ser computacionalmente menos costoso realizar una codificación CABAC de derivación en un bin que realizar una codificación CABAC regular en el bin. Además, llevar a cabo la codificación CABAC de derivación puede permitir un mayor grado de paralelización y rendimiento. Los bins codificados usando la codificación CABAC de derivación pueden denominarse "bins de derivación". El motor de codificación CABAC de derivación puede ser más simple porque el motor de codificación CABAC de derivación no selecciona contextos y puede asumir una probabilidad de ½ para ambos símbolos (0 y 1). En consecuencia, al obtener la codificación CABAC, los intervalos se dividen directamente a la mitad. Agrupar bins de derivación juntos puede incrementar el rendimiento del codificador de vídeo 20 y del descodificador de vídeo 30. El motor de codificación CABAC de derivación puede codificar varios bins en un solo ciclo, mientras que el motor de codificación CABAC normal puede codificar solo un único bin en un ciclo.

[0051] Por tanto, la codificación aritmética puede basarse en la división recursiva por intervalos. En un codificador aritmético convencional, un rango, con un valor inicial de 0 a 1, se divide en dos subintervalos en base a la probabilidad del bin. Los bits codificados proporcionan un desplazamiento que, cuando se convierte en una fracción binaria, selecciona uno de los dos subintervalos, lo que indica el valor del bin descodificado. Después de cada bin descodificado, el rango se actualiza para igualar el subintervalo seleccionado, y el proceso de división de intervalo se repite por sí mismo. El rango y el desplazamiento tienen una precisión de bits limitada, por lo que se puede requerir una nueva normalización siempre que el rango caiga por debajo de un determinado valor para evitar el flujo inferior. La renormalización puede ocurrir después de descodificar cada bin.

[0052] La codificación aritmética se puede hacer usando una probabilidad estimada (codificada por contexto) (que se denomina modo de codificación regular), o suponiendo una probabilidad igual de 0,5 (codificación de derivación, que se denomina modo de derivación). Para los bins de derivación codificados, la división del rango en subintervalos puede realizarse mediante una conmutación, mientras que puede ser necesaria una tabla de búsqueda para los bins codificados por contexto. HEVC usa la misma codificación aritmética que H.264/AVC. Un ejemplo del proceso de descodificación se describe en la sección 9.3.4.3.2.2 de JCTVC-L1003.

[0053] Se usan varios procesos de binarización diferentes en HEVC, incluyendo unario (U), unario truncado (TU), Exp-Golomb de K-ésimo orden (EGk) y longitud fija (FL). Los detalles de los procesos de binarización usados en HEVC se pueden encontrar en JCTVC-L1003. Como se menciona anteriormente, la codificación de Exp-Golomb de K-ésimo orden es uno de los procesos de binarización usados en HEVC. Cuando el orden K es igual a 0, se puede usar el siguiente proceso para generar una cadena bin (por ejemplo, una cadena de bins) usando codificación Exp-Golomb para un número entero x no negativo dado:

1. Se registra x+1 en binario.

2. Se cuentan los bits escritos, se resta uno y se registra ese número de bits que empiezan en cero que preceden a la cadena de bits anterior (es decir, se registra el número de bits en la representación binaria de *x*+1, menos 1, antepuesta a la representación binaria de *x*+1).

Por ejemplo, siendo *x* igual a 4. Por tanto, en la etapa 1, se registra el valor binario 101. En la etapa 2, el número de bits es 3; 3 menos 1 es 2. Por consiguiente, el prefijo consiste en dos 0. Por lo tanto, el código Exp-Golomb de K-ésimo orden para 4 es 00 101.

[0054] Para codificar un número entero x no negativo en un código de orden k Exp-Golomb:

1. Codificar $\lfloor x/2^k \rfloor$ usando el código de orden 0 Exp-Golomb descrito anteriormente, a continuación

2. Codificar $x \bmod 2^k$ en un valor binario de longitud fija de k bits.

Una forma equivalente de expresar esto es:

1. Codificar $x+2^k-1$ usando el código de orden 0 Exp-Golomb (es decir, se codifica $x+2^k$) usando un código gamma de Elias), a continuación

2. Eliminar los bits cero que dan como resultado *k* del resultado de codificación.

Por ejemplo, siendo K igual a 2 y *x* igual a 4. En este ejemplo, en la etapa 1, $\lfloor 4/2^2 \rfloor$ es igual a 1. El código Exp-Golomb de 0-ésimo orden para 1 es 010. En la etapa 2, 4 mod 2^2 es 0. La versión codificada de 0 es un valor de longitud fija de 2 bits que es 00. Este valor de longitud fija se agrega al código Exp-Golomb de 0-ésimo orden descrito previamente. Por tanto, el código Exp-Golomb de 2.º orden final para 4 es 010 00.

[0055] En la tabla a continuación se da un ejemplo:

Tabla 1. Palabra de código de m usando el código Exp-Golomb de K-ésimo orden

Valor m	***k*=0**	***k*=1**	***k*=2**	***k*=3**
0	1	10	100	1000
1	010	11	101	1001
2	011	0100	110	1010
3	00100	0101	111	1011
4	00101	0110	01000	1100
5	00110	0111	01001	1101
6	00111	001000	01010	1110
7	0001000	001001	01011	1111
8	0001001	001010	01100	010000
9	0001010	001011	01101	010001

En otra forma de representación:

Tabla 2. Palabra de código de m usando el código Exp-Golomb de K-ésimo orden

Valor m	***k*=0**	***Valor m***	***k*=1**	***Valor m***	***k*=2**
0	1	0-1	1X	0-3	1XX
1-2	01X	2-5	01XX	4-11	01XXX
3-6	001XX	6-13	001XXX	12-27	001XXXX
7-14	0001XXX	14-29	0001XXXX	28-59	0001XXXXX
15-30	00001XXXX	30-61	00001XXXXX	60-123	00001XXXXXX
..	.....	..	.....	..	.....

[0056] En la Tabla 2, el valor de X podría ser 0 o 1. Debe tenerse en cuenta que para algunos casos, '1' y '0' enumerados anteriormente podrían intercambiarse. Es decir, la cadena de bin puede estar dirigida por '1' consecutivos en lugar de '0'.

[0057] Para un codificador de Rice de K-ésimo orden, se puede generar una palabra de código mediante las siguientes etapas:

1. Fijar el parámetro *M* a un valor entero (M = (1<<K).
2. Para *N,* el número que se va a codificar, encontrar
    1. cociente = *q* = int[*N*/*M*]
    2. resto = *r* = *N* módulo *M*
3. Generar palabra de código
    1. El formato del código: <código de cociente><código de resto>, donde
    2. Código de cociente (en codificación unaria)
        1. Escribir una cadena de longitud *q* de 1 bits
        2. Escribir un bit 0
    3. Código de resto (en codificación binaria truncada): Se necesitan K bits

[0058] Por ejemplo, sea K igual a 2 y *N* igual a 5. Por tanto, M es igual a 100 en binario (4 en decimal). En este ejemplo, el cociente de división entera *q* de 5/4 es igual a 1; el resto es igual a 1 (1 en binario). Por tanto, el código del cociente es 10 (es decir, una cadena de longitud 1 de bits 1, seguida de un bit 0). El código del resto es 01 (es decir, un código de longitud fija de K bits que representa el valor de resto 1). Por lo tanto, el código Rice de 2.º orden para 5 es 1001.

[0059] Los códigos Rice con divisor M igual a 4, para números hasta 15, se muestran en la Tabla 3, a continuación:

Tabla 3. Palabra de código de m usando código Rice de K-ésimo orden (K igual a 2)

Valor	Cociente	Resto	Código
0	0	0	0 00
1	0	1	0 01
2	0	2	0 10
3	0	3	0 11
4	1	0	1 0 00
5	1	1	1 0 01
6	1	2	1 0 10
7	1	3	1 0 11
8	2	0	11 0 00
9	2	1	11 0 01
10	2	2	11 1 10
11	2	3	11 0 11
12	3	0	111 0 00
13	3	1	111 0 01
14	3	2	111 0 10
15	3	3	111 0 11

[0060] Como se describe brevemente anteriormente, el proceso CABAC usado en HEVC y potencialmente en otras especificaciones de codificación de vídeo usa el modelado de contexto. El modelado de contexto proporciona una estimación de probabilidad exacta que es necesaria para lograr una alta eficacia de codificación. En consecuencia, el modelado de contexto es altamente adaptativo. Se pueden usar diferentes modelos de contexto para diferentes bins donde la probabilidad de los modelos de contexto se actualiza en base a los valores de bins codificados previamente. Los bins con distribuciones similares a menudo comparten el mismo modelo de contexto. El modelo de contexto para cada bin puede seleccionarse en base al tipo de elemento sintáctico, la posición del bin en el elemento sintáctico (binIdx), luma/croma, información vecina, etc.

[0061] En algunos ejemplos, el cambio de contexto se produce después de cada codificación de bin. Además, en algunos ejemplos, los modelos de probabilidad se almacenan como entradas de 7 bits (6 bits para el estado de probabilidad y 1 bit para el símbolo más probable (MPS)) en la memoria de contexto y se abordan usando el índice de contexto calculado según la lógica de selección de contexto.

[0062] Como se describe brevemente anteriormente, en HEVC, una CU puede dividirse en TU de acuerdo con un árbol cuaternario residual. Por tanto, para adaptar las diversas características de los bloques residuales, se aplica una estructura de codificación de transformada usando el árbol cuaternario residual (RQT) en HEVC, que se describe brevemente en el artículo "Transform Coding Using the Residual Quadtree (RQT)", Fraunhofer HHI, que anteriormente estaba disponible en http://www.hhi.fraunhofer.de/fields-of-competence/image-processing/researchgroups/image-video-coding/hevc-high-efficiency-video-coding/transform-coding-using-the-residual-quadtree-rqt.html, pero que a partir del 28 de junio de 2016 está disponible en http://www.hhi.fraunhofer.de/departments/video-coding-analytics/research-groups/image-video-coding/research-topics/hevc-high-efficiency-video-coding/transform-coding-using-the-residual-quadtree-rqt.html.

[0063] Como se describe anteriormente, en HEVC, cada imagen se divide en unidades de árbol de codificación (CTU), que se codifican en orden de escaneo por trama para un mosaico o fragmento específico. Una CTU es un bloque cuadrado y representa la raíz de un árbol cuaternario, es decir, el árbol de codificación. El tamaño de la CTU puede variar de muestras de luma de 8×8 a 64×64, pero típicamente se usa 64×64. Cada CTU se puede dividir además en bloques cuadrados más pequeños llamados unidades de codificación (CU). Después de que la CTU se divide de forma recursiva en CU, cada CU se divide además en PU y TU. La partición de una CU en TU se lleva a cabo de forma recursiva en base a un enfoque de árbol cuaternario, por lo tanto, la señal residual de cada CU está codificada por una estructura de árbol, a saber, el RQT. El RQT permite tamaños de TU de muestras de luma de 4×4 hasta 32×32.

[0064] La FIG. 2 es un diagrama conceptual que ilustra un esquema de transformada basado en un árbol cuaternario residual, tal como el usado en la codificación de vídeo de alta eficacia (HEVC). La FIG. 2 muestra un ejemplo donde una CU incluye diez TU, marcadas con las letras a a j, y la división de bloques correspondiente. Cada nodo del RQT es en realidad una TU. Las TU individuales se procesan en orden de transversal de árbol de primera profundidad, que se ilustra en la figura como orden alfabético, que sigue a un escaneo en Z recursivo con la transversal de primera profundidad. El enfoque de árbol cuaternario permite la adaptación de la transformada a las características de frecuencia espacial variables de la señal residual. Típicamente, los tamaños de bloque de transformada más grandes, que tienen mayor soporte espacial, proporcionan una mejor resolución de frecuencia. Sin embargo, los tamaños de bloque de transformada más pequeños, que tienen menor soporte espacial, proporcionan una mejor resolución espacial. La compensación entre las dos resoluciones, espacial y de frecuencia, se puede seleccionar mediante la decisión del modo del codificador, por ejemplo, en base a la técnica de optimización de la distorsión de velocidad. La técnica de optimización de distorsión de velocidad calcula una suma ponderada de bits de codificación y distorsión de reconstrucción, es decir, el coste de distorsión de velocidad, para cada modo de codificación (por ejemplo, una estructura de división de RQT específica), y selecciona el modo de codificación con el menor coste de distorsión de velocidad como el mejor modo.

[0065] En algunos ejemplos, se definen tres parámetros en el RQT: (1) la profundidad máxima del árbol, (2) el tamaño de transformada mínimo permitido y (3) el tamaño de transformada máximo permitido. En la HEVC, los tamaños de transformada mínimos y máximos pueden variar dentro del rango de muestras de 4×4 a 32×32, que corresponden a las transformadas de bloque soportadas mencionadas anteriormente. La profundidad máxima permitida del RQT restringe el número de TU. Una profundidad máxima igual a 1 significa que una CU no puede dividirse más si cada TU incluida alcanza el tamaño de transformada máximo permitido, por ejemplo, 32x32.

[0066] Todos estos parámetros pueden interactuar e influir en la estructura de RQT. Por ejemplo, se considera un caso en el que el tamaño de CU de raíz es 64×64, la profundidad máxima es igual a cero y el tamaño máximo de transformada es igual a 32×32. En este caso, la CU tiene que dividirse al menos una vez, ya que de lo contrario daría lugar a una TU de 64×64, lo cual no está permitido. Los parámetros de RQT, es decir, la profundidad máxima de RQT, el tamaño de transformada mínimo y máximo, pueden transmitirse en el flujo de bits al nivel del conjunto de parámetros de secuencia. Con respecto a la profundidad de RQT, se pueden especificar y señalar diferentes valores para las CU intra e intercodificadas. Por ejemplo, se puede especificar un primer valor de profundidad máxima para las CU codificadas en modo intra y un segundo valor de profundidad máxima diferente para las CU codificadas en modo inter.

[0067] La transformada de árbol cuaternario se aplica para bloques tanto intra como interresiduales. Típicamente, se aplica una transformada DCT-II (es decir, una transformada de coseno discreto de tipo 2) del mismo tamaño de la división de árbol cuaternario residual actual para un bloque residual. Sin embargo, si el bloque de árbol cuaternario residual actual es de 4×4 y se genera por intrapredicción, se aplica una transformada DST-VII de 4×4 (es decir, una transformada de seno discreto de tipo 7). En la HEVC, las transformadas de mayor tamaño, por ejemplo, las transformadas de 64×64, no se adoptan principalmente debido a su beneficio limitado y complejidad relativamente alta para vídeos de resolución relativamente inferiores.

[0068] En la HEVC, independientemente del tamaño de la TU, el residuo de una TU se codifica con grupos de coeficientes (CG) no superpuestos. Cada uno de los CG contiene los coeficientes de un bloque de 4×4 de una TU. Por ejemplo, una TU de 32×32 tiene 64 CG en total, y una TU de 16×16 tiene 16 CG en total. Los CG dentro de una TU se codifican de acuerdo con un determinado orden de escaneo predefinido. Al codificar cada CG respectivo, los coeficientes dentro del CG respectivo se escanean y codifican de acuerdo con un determinado orden de escaneo predefinido para un bloque de 4×4. La FIG. 3 ilustra el escaneo de coeficientes para una TU de 8×8 que contiene 4 CG usados en la HEVC.

[0069] La tabla de elementos sintácticos usada en la HEVC para señalizar datos residuales de una TU se define como sigue:

7.3.8.11 Sintaxis de codificación residual

[0070]

residual_coding(x0, y0, log2TrafoSize, cIdx) {	**Descriptor**
if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize == 2))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize − 2)) * (1 << (log2TrafoSize − 2)) − 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = 16	
lastSubBlock− −	
}	
lastScanPos− −	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	

yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i− −) {	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][i][1]	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
for(n = (i == lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
}	
firstSigScanPos = 16	
lastSigScanPos = −1	
numGreater1Flag = 0	
lastGreater1ScanPos = −1	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	

if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos == −1)	
lastGreater1ScanPos = n	
}	
if(lastSigScanPos == −1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	
signHidden = (lastSigScanPos − firstSigScanPos > 3 && !cu_transquant_bypass_flag)	
if(lastGreater1ScanPos != −1)	
coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC] && (!sign_data_hiding_enabled_flag || !signHidden || (n != firstSigScanPos)))	
coeff_sign_flag[n]	ae(v)
}	
numSigCoeff = 0	
sumAbsLevel = 0	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	

baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel == ((numSigCoeff < 8) ? ((n == lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])	
if(sign_data_hiding_enabled_flag && signHidden) {	
sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
if((n == firstSigScanPos) && ((sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
numSigCoeff++	
}	
}	
}	
}	

[0071] Para cada componente de color respectivo (por ejemplo, luma, Cb, Cr), se puede señalizar en primer lugar un indicador respectivo para indicar si una TU actual tiene al menos un coeficiente distinto de cero. Si hay al menos un coeficiente distinto de cero, la posición del último coeficiente significativo en el orden de escaneo del coeficiente en la TU se codifica explícitamente a continuación con una coordenada relativa a la esquina superior izquierda de la TU. El componente vertical u horizontal de la coordenada está representado por su prefijo y sufijo. El prefijo se binariza con Rice truncado (TR) de 0-ésimo orden y el sufijo se binariza con longitud fija. El análisis de la codificación de Rice anterior es un caso de codificación de Rice truncada (TR).

[0072] En la tabla anterior, **last_sig_coeff_x_prefix** especifica el prefijo de la posición de la columna del último coeficiente significativo en el orden de escaneo dentro de un bloque de transformada. El valor de last_sig_coeff_x_prefix estará dentro del rango de 0 a (log2TrafoSize << 1) – 1, inclusive.

[0073] En la tabla anterior, **last_sig_coeff_y_prefix** especifica el prefijo de la posición de la fila del último coeficiente significativo en el orden de escaneo dentro de un bloque de transformada. El valor de last_sig_coeff_y_prefix estará dentro del rango de 0 a (log2TrafoSize << 1) – 1, inclusive.

[0074] En la tabla anterior, **last_sig_coeff_x_suffix** especifica el sufijo de la posición de la columna del último coeficiente significativo en el orden de escaneo dentro de un bloque de transformada. Los valores de last_sig_coeff_x_suffix estarán en el rango de 0 a (1 << ((last sig_coeff_x_prefix >> 1) – 1)) – 1, inclusive. La posición de la columna del último coeficiente significativo en el orden de escaneo dentro de un bloque de transformada LastSignificantCoeffX se deriva como sigue:

- Si last_sig_coeff_x_suffix no está presente, se aplica lo siguiente:

LastSignificantCoeffX = last_sig_coeff_x_prefix

- Si de lo contrario (last_sig_coeff_x_suffix está presente), se aplica lo siguiente:

LastSignificantCoeffX = (1 << ((last_sig_coeff_x_prefix >> 1) – 1)) * (2 + (last_sig_coeff_x_prefix & 1)) + last_sig_coeff_x_suffix

[0075] En la tabla anterior, **last_sig_coeff_y_suffix** especifica el sufijo de la posición de la fila del último coeficiente significativo en el orden de escaneo dentro de un bloque de transformada. El valor de last_sig_coeff_y_suffix estará en el rango de 0 a (1 << ((last_sig_coeff_y_prefix >> 1) – 1)) – 1, inclusive.

La posición de la fila del último coeficiente significativo en el orden de escaneo dentro de un bloque de transformada LastSignificantCoeffY se deriva como sigue:

- Si last_sig_coeff_y_suffix no está presente, se aplica lo siguiente:

LastSignificantCoeffY = last_sig_coeff_y_prefix

- Si de lo contrario (last_sig_coeff_ y_suffix está presente), se aplica lo siguiente:

LastSignificantCoeffY = (1 << ((last_sig_coeff_y_prefix >> 1) – 1)) * (2 + (last_sig_coeff_y_prefix & 1)) + last_sig_coeff_y_suffix

Cuando scanIdx es igual a 2, las coordenadas se intercambian como sigue:

(LastSignificantCoeffX, LastSignificantCoeffY)=Swap(LastSignificantCoeffX, LastSignificantCoeffY).

[0076] Con dicha posición codificada y también el orden de escaneo de coeficientes de los CG, un indicador se señaliza además para CG excepto el último CG (en orden de escaneo) que indica si el último CG contiene coeficientes distintos de cero.

[0077] Al codificar si un CG tiene coeficientes distintos de cero, es decir, el indicador de CG (**coded_sub_block_flag** en la especificación de HEVC), un codificador de vídeo puede usar la información de los CG vecinos para construir el contexto. Por ejemplo, en HEVC, la selección de contexto para codificar el indicador de CG se define como:

(CG derecho disponible && Indicador de CG derecho es igual a 1) || (CG inferior disponible && Indicador de CG inferior es igual a 1) En la fórmula anterior, CG derecho y CG inferior son los dos CG vecinos cercanos al CG actual. Por ejemplo, en la FIG. 3, al codificar el bloque de 4×4 superior izquierdo, el CG derecho se define como el bloque de 4×4 superior derecho y el CG inferior se define como el bloque de 4×4 inferior izquierdo. Croma y luma usan diferentes conjuntos de modelos de contexto, pero un codificador de vídeo puede usar la misma regla para seleccionar uno de los modelos de contexto para cada uno de croma y luma. Los detalles de la derivación del incremento del índice de contexto (ctxInc) se pueden encontrar en la subcláusula 9.3.4.2.4 de la especificación de HEVC.

[0078] Para aquellos CG que contienen coeficientes distintos de cero, un codificador de vídeo puede codificar además indicadores significativos (significant_flag o sig_coeff_flag), valores absolutos de coeficientes (incluyendo coeff_abs _level_greater1_flag, coeff_abs_level_greater2_flag y coeff_abs_level_remaining) y señalizar información (coeff sign flag) para cada coeficiente de acuerdo con el orden de escaneo del coeficiente de 4x4 predefinido. La codificación de los niveles del coeficiente de transformada se separa en múltiples pases de escaneo.

[0079] En el primer pase de la primera codificación de bin, todos los primeros bins (o el índice de bin 0, **bin0**, que puede corresponder a elementos sintácticos de indicadores significativos) de coeficientes de transformada en cada posición dentro de un CG están codificados, excepto que podría deducirse que el coeficiente de transformada específico es igual a 0. Por ejemplo, para el último CG que contiene el coeficiente en la última posición, todos los coeficientes que se escanean antes de la última posición en el orden de descodificación se derivan a 0 y no se codifican bins.

[0080] Como se describe en la subcláusula 9.3.4.2.5 de la especificación de HEVC, se usa una variable sigCtx para derivar el valor de una variable de incremento de índice de contexto (ctxInc) usada en la selección del contexto de codificación para un indicador significativo. La variable sigCtx depende de la localización actual en relación con la posición superior izquierda de la TU actual, el índice de componente de color cIdx, el tamaño del bloque de transformada y los bins descodificados previamente del elemento sintáctico coded_sub_block_flag. Se aplican diferentes reglas dependiendo del tamaño de la TU. La subcláusula 9.3.4.2.5 de la especificación de HEVC proporciona detalles de la selección del incremento del índice de contexto para el indicador significativo **Error! Reference source not found.. [¡Error! Fuente de referencia no encontrada..]**

[0081] En el segundo pase de la segunda codificación de bin, un codificador de vídeo codifica coeff_abs_level_greater1_flags. El modelado de contexto para coeff_abs_level_greater1_flags depende de un índice de componente de color, un índice de escaneo de subbloque actual y un índice de escaneo de coeficiente actual dentro del subbloque actual. La subcláusula 9.3.4.2.6 de la especificación de HEVC proporciona detalles de la selección de la variable de incremento del índice de contexto para coeff_abs_level_greater1_flags.

[0082] En el tercer pase de la tercera codificación de bin, el codificador de vídeo codifica coeff_abs_level_greater2_flags. El modelado de contexto para coeff_abs_level_greater2_flags es similar al usado por coeff_abs_level_greater1_flags. La subcláusula 9.3.4.2.7 de la especificación de HEVC proporciona detalles de la selección del incremento del índice de contexto.

[0083] Para mejorar el rendimiento, el segundo y tercer pase pueden no procesar todos los coeficientes en un CG. Por el contrario, los primeros ocho coeff_abs_level_greater1_flags en un CG están codificados en modo regular. Después de eso, los valores se dejan codificar en modo de derivación en el quinto pase por la sintaxis coeff_abs_level_remaining. De forma similar, solo se codifica el coeff_abs_level_greater2_flag para el primer coeficiente en un CG con una magnitud mayor que 1. El resto de coeficientes con una magnitud mayor que 1 del CG usan coeff_abs_level_remaining para codificar el valor. Esta técnica puede limitar el número de bins regulares para niveles de coeficientes a un máximo de 9 por CG: 8 para el coeff_abs_level_greater1_flags y 1 para coeff_abs_level_greater2_flags.

[0084] El cuarto pase es para información de signos. En la HEVC, el signo de cada coeficiente distinto de cero se codifica en el cuarto pase de escaneo en modo de derivación. Para cada CG, y dependiendo de un criterio, la codificación del signo del último coeficiente distinto de cero (en orden de escaneo inverso) simplemente se omite cuando se usa el ocultamiento de datos de signos (SDH). En cambio, el valor del signo se incrusta en la paridad de la suma de los niveles del CG usando una convención predefinida: incluso corresponde a "+" y raramente a "-". El criterio para usar SDH es la distancia en el orden de escaneo entre el primer y el último coeficiente distinto de cero del CG. En particular, en HEVC, si esta distancia es igual o mayor que 4, se usa SDH. De lo contrario, no se usa SDH. El valor de 4 se seleccionó en HEVC porque el valor de 4 proporciona la mayor ganancia en las secuencias de prueba de HEVC.

[0085] En el último pase, los bins restantes se codifican en un pase de escaneo adicional. Sea que el *baseLevel* de un coeficiente se defina como:

$$baseLevel = \text{significant\_flag} + \text{coeff\_abs\_level\_greater1\_flag} + \text{coeff\_abs\_l evel\_greater2\_flag} \quad (1)$$

donde un indicador (es decir, indicador significativo, coeff_abs_level_greater1_flag y coeff_abs_level_greater2_flag) tiene un valor de 0 o 1 y se infiere que es 0 si no está presente. A continuación, el valor absoluto del coeficiente se define simplemente como:

$$\text{absCoeffLevel} = baseLevel + \text{coeff\_abs\_level\_remaining}. \quad (2)$$

En la HEVC, coeff_abs_level_remaining está codificado por derivación y, por lo tanto, no se necesita modelado de contexto.

[0086] El elemento sintáctico *coeff_abs_level_remaining* en HEVC indica el valor restante para el valor absoluto de un nivel de coeficiente (si el valor es mayor que el codificado en pases de escaneo previos para la codificación del coeficiente). El elemento sintáctico *coeff_abs_level_remaining* está codificado en modo de derivación para incrementar el rendimiento. Como se describe en Chien *et al.*, "On Coefficient Level Remaining Coding", JCTVC-I0487, 9.ª Reunión del Equipo de Colaboración Conjunta sobre Codificación de Vídeo (JCT-VC), Ginebra, Suiza, abril-mayo de 2012, HEVC emplea códigos Rice para pequeños valores de *coeff_abs_level_remaining* y cambia a un código Exp-Golomb para valores mayores de *coeff_abs_level_remaining.*

[0087] El punto en el que HEVC cambia de usar códigos Rice para elementos sintácticos *coeff_abs_level_remaining* a usar códigos Exp-Golomb para elementos sintácticos *coeff_abs_level_remaining* puede denominarse punto de conmutación. El punto de conmutación se puede definir como uno de los dos procedimientos a continuación:

1) el punto de conmutación es igual a (3<<K): cuando coeff_abs_level_remaining es menor que (3<<K), se usa un código Rice de K-ésimo orden. De lo contrario, se usa un prefijo (con tres '1') y un sufijo que usa un código Exp-Golomb de K-ésimo orden para la codificación de derivación

2) el punto de conmutación es igual a (4<<K): cuando coeff_abs_level_remaining es menor que (4<<K), se usa un código Rice de K-ésimo orden. De lo contrario, se usa un prefijo (con cuatro '1') y un sufijo que usa un código Exp-Golomb de (K+1)-ésimo orden para la codificación de derivación.

[0088] Para simplificar, en las descripciones a continuación, se usa el primer procedimiento, es decir, el punto de conmutación se define como (3<<K). Se da un ejemplo en la Tabla 4:

Tabla 4. Palabra de código de *coeff_abs_level_remaining* igual a m

Valor m	***k*=0**		***Valor m***	*k*=1	
	Prefijo	***Sufijo***		***Prefijo***	***Sufijo***
0	0		0-1	0X	
1	10		2-3	10X	
2	110		4-5	110X	
3	111	0	6-7	111	0X
4-5	111	10X	8-11	111	10XX
6-9	111	110XX	12-19	111	110XXX
10-17	111	1110XXX	20-35	111	1110XXXX
...	...	...	...	...	....

[0089] Como se describe en J. Sole *et al*., "Transform Coefficient Coding in HEVC", Transacciones del IEEE en circuitos y sistemas para transmisión de vídeo (número especial en HEVC), diciembre de 2012), en HEVC, el parámetro K de Rice se establece en 0 al comienzo de cada CG (grupo de coeficientes, que es un subbloque de 4x4) y K se actualiza condicionalmente dependiendo del valor previo del parámetro y el nivel absoluto actual como sigue:

$$\text{Si absCoeffLevel} > 3*2K, \text{ entonces } K = \min(K+1, 4) \qquad (3)$$

$$\text{De lo contrario, } K = K$$

donde K es el parámetro de Rice y la función min() devuelve el valor inferior entre dos entradas. Por tanto, si el valor *absCoeffLevel* para un nivel de coeficiente actual es mayor que $3*2^K$, un codificador de vídeo actualiza K para que sea igual a cualquiera que sea el menor de K+1 y 4. Si el valor *absCoeffLevel* del nivel del coeficiente actual no es mayor que $3*2^K$, el codificador de vídeo no actualiza K. El proceso de actualización de parámetros permite que la binarización se adapte a las estadísticas del coeficiente cuando se observan valores grandes en la distribución. En las siguientes descripciones, dicho procedimiento de derivación de un parámetro de Rice se denomina 'Procedimiento de derivación basado en retrospectiva'.

[0090] Otra técnica, propuesta en Karczewicz y *et al*., "RCE2: Results of Test 1 on Rice Parameter Initialization", JCTVC-P0199, Equipo de Colaboración Conjunta sobre Codificación de Vídeo (JCT-VC) de la ITU-T SG 16 WP 3 e ISO/IEC JTC 1/SC 29/WG 11, 16.ª Reunión: San José, EE. UU., 9-17 de enero de 2014 (más adelante en el presente documento, "JCTVC-P0199"), se emplea para la derivación del parámetro de Rice en Extensiones de rango de HEVC para codificar más eficazmente los niveles de coeficientes mucho más grandes que pueden estar presentes en codificación de bit de alta profundidad o sin pérdida.

[0091] En primer lugar, en la técnica usada en las extensiones de rango de HEVC, los subbloques de 4 × 4 (es decir, CG) se dividen en diferentes categorías ("sbType"). Para cada subbloque, se deriva un parámetro de Rice inicial basado en subbloques previamente codificados en la misma categoría. La categorización se basa en si el subbloque (es decir, CG) es un bloque de omisión de transformada ("isTSFlag") y si el bloque es para el componente de luma:

$$\text{sbType} = \text{isLuma}*2 + \text{isTSFlag} \qquad (4)$$

Un codificador de vídeo mantiene estadísticas *statCoeff* para cada tipo de subbloque (sbType) dependiendo del *coeff_abs_level_remaining* del primer coeficiente en el subbloque:

$$\text{si (absCoeffLevel} >= 3 * (1 << (\text{statCoeff}/4))) \quad \text{statCoeff}++; \qquad (5)$$

$$\text{si no ((2 * absCoeffLevel)} < (1 << (\text{statCoeff}/4))) \quad \text{statCoeff}--;$$

El codificador de vídeo actualiza la variable statCoeff que se actualiza como máximo una vez por cada subbloque de 4×4 usando el valor del primer *coeff_abs_level_remaining* codificado del subbloque. El codificador de vídeo restablece las entradas de *statCoeff* a 0 al comienzo del fragmento. Además, el codificador de vídeo usa el valor de *statCoeff* para inicializar el parámetro *K de* Rice al comienzo de cada subbloque de 4×4 como:

$$\text{cRiceParam} = \text{Min}(\text{maxRicePara}, \text{statCoeff}/4). \qquad (6)$$

[0092] En las siguientes descripciones, dicho procedimiento para derivar un parámetro de Rice (por ejemplo,

cRiceParam) se denomina "procedimiento de derivación basado en estadísticas" o "procedimiento de derivación de parámetro de Rice basado en estadísticas". Por tanto, en el procedimiento de derivación basado en estadísticas, un codificador de vídeo puede dividir la pluralidad de subbloques de una imagen actual en una pluralidad de categorías de modo que, para subbloques respectivos de la pluralidad de subbloques, el subbloque respectivo se clasifica en base a si el subbloque respectivo es un bloque de omisión de transformada y si el subbloque respectivo es para un componente de luma. Además, para cada categoría respectiva de la pluralidad de categorías, el codificador de vídeo puede mantener un valor estadístico respectivo para la categoría respectiva. Para cada subbloque respectivo de la pluralidad de subbloques, el codificador de vídeo puede usar el valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo para inicializar un parámetro de Rice respectivo para el subbloque respectivo. Un elemento sintáctico *coeff_abs_level_remaining* en un subbloque particular de la pluralidad de subbloques se binariza usando un código Rice de K-ésimo orden, siendo K el parámetro de Rice para el subbloque particular.

[0093] Para cada categoría respectiva de la pluralidad de categorías, como parte del mantenimiento del valor estadístico respectivo para la categoría respectiva, el codificador de vídeo puede, para cada subbloque respectivo de la imagen que pertenece a la categoría respectiva, actualizar el valor estadístico respectivo para la categoría respectiva como máximo una vez para el subbloque respectivo usando un elemento sintáctico de nivel restante codificado primero para el subbloque respectivo. Además, como parte de la actualización del valor estadístico respectivo para la categoría respectiva, el codificador de vídeo puede incrementar el valor estadístico respectivo para la categoría respectiva si (absCoeffLevel >=3*(1<<(statCoeff /4))); y disminuir el valor estadístico respectivo para la categoría respectiva si ((2*absCoeffLevel)<(1<<(statCoeff/4))). Como antes, absCoeffLevel es un nivel de coeficiente absoluto del subbloque respectivo y statCoeff es el valor estadístico respectivo para la categoría respectiva. Para cada subbloque respectivo de la pluralidad de subbloques, como parte del uso del valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo para inicializar un parámetro de Rice respectivo para el subbloque respectivo, el codificador de vídeo puede determinar el parámetro respectivo de Rice para el subbloque respectivo como mínimo de un parámetro máximo de Rice y el valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo dividido por 4.

[0094] Nguyen *et al.*, "Non-CEll: Proposed Cleanup for Transform Coefficient Coding", JCTVC-H0228, 8.ª Reunión: San José, CA, EE. UU., 1-10 de febrero de 2012 (más adelante en el presente documento, "JCTVC-H0228") propusieron la codificación de pase de un escaneo, es decir, toda la información sobre un nivel de coeficiente de transformada se codifica en una sola etapa en lugar de la codificación de múltiples pases como en HEVC. Para cada posición de escaneo, se evalúan los vecinos (es decir, las muestras vecinas) cubiertos por una plantilla local, como se hace para bin0 (el primer bin de la cadena bin, también conocido como significant_coeff_flag o coeff_abs_greater0_flag) en el diseño actual de HEVC. De esta evaluación, se derivan los modelos de contexto y el parámetro de Rice, que controla la binarización adaptativa del valor absoluto restante. Para ser más específicos, los modelos de contexto para los parámetros bin0, bin1, bin2 y de Rice están todos seleccionados (bin1 y bin2 también se denominan coeff_abs_greater1_flag y coeff_abs_greater2_flag) en base a las magnitudes de los coeficientes de transformada localizados en la plantilla local.

[0095] La FIG. 4 es un diagrama conceptual que ilustra una plantilla local ejemplar. En particular, se da un ejemplo para una plantilla local en la FIG. 4 para un bloque de transformada de 8×8 con escaneo diagonal, **x** indica la posición de escaneo actual y x_i con $i \in [0,4]$ indica los vecinos cubiertos por la plantilla local.

[0096] En las siguientes ecuaciones, *sum_absolute_level,* que indica la suma absoluta de los vecinos, y *sum_absolute_levelMinus1*, que indica la suma absoluta de cada nivel menos 1, se usan para derivar índices de contexto para **bin0, bin1, bin2** (es decir, para un indicador significativo, coeff_abs_greater1_flag y coeff_abs_greater2_flag), y para determinar el parámetro de Rice r.

$$sum\_absolute\_level = \sum |x_i|$$

$$sum\_absolute\_levelMinus1 = \sum \delta_j(x_i) \qquad (7)$$

$$\delta_j(x) = \begin{cases} |x_i| - 1 & |x_i| > 0 \\ 0 & x_i = 0 \end{cases}$$

con

[0097] En la técnica descrita en JCTVC-H0228, el parámetro de Rice r se deriva como sigue. Para cada posición de escaneo, el parámetro se establece en 0. A continuación, el *sum_absolute_levelMinus1* se compara con un conjunto de umbrales $t_R = \{3, 9, 21\}$. En otras palabras, el parámetro de Rice es 0 si la *sum_absolute_levelminus1* cae en el primer intervalo, es 1 si *sum_absolute_levelMinus1* cae en el segundo intervalo y así sucesivamente. La derivación del parámetro r de Rice se resume a continuación.

$$r(x)=\begin{cases}0 & x\in[0,3]\\ 1 & x\in[4,9]\\ 2 & x\in[10,21]\\ 3 & x>21\end{cases} \qquad (8)$$

con *x* igual a *sum_absolute_levelMinus1.* El rango del parámetro de Rice está dentro de [0, 3].

[0098] En las siguientes descripciones, dicho procedimiento de derivación del parámetro de Rice se denomina "Procedimiento de derivación basado en plantillas" o "Procedimiento de derivación de parámetro de Rice basado en plantillas". Por tanto, en el procedimiento de derivación basado en plantillas, una imagen actual comprende una pluralidad de subbloques de 4x4, una plantilla local cubre los vecinos de una muestra actual de la TU, y la muestra actual está en una posición de escaneo actual. Además, el codificador de vídeo 20 puede señalizar y el descodificador de vídeo 30 puede obtener un elemento sintáctico (por ejemplo, un elemento sintáctico *coeff_abs_level_remaining*) que indica un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo para la muestra actual. En el procedimiento de derivación basado en plantillas, para cada vecino respectivo cubierto por la plantilla local, un codificador de vídeo (por ejemplo, codificador de vídeo 20 o descodificador de vídeo 30) puede determinar un valor respectivo para el vecino respectivo. En este ejemplo, el valor respectivo para el vecino respectivo es igual al valor absoluto del vecino menos 1 si el valor absoluto del vecino es mayor que 0 e igual a 0 si el vecino es igual a 0. Además, en el procedimiento de derivación basado en plantillas, el codificador de vídeo puede determinar un valor de suma (por ejemplo, *sum_absolute_levelMinus1*) igual a una suma de valores para los vecinos. Además, en el procedimiento de derivación basado en plantillas, el codificador de vídeo puede determinar que un parámetro de Rice es igual a 0 si el valor de la suma cae en un primer intervalo (por ejemplo, x es un número entero entre 0 y 3, inclusive), igual a 1 si el valor de la suma cae en un segundo intervalo (por ejemplo, x es un número entero entre 4 y 9, inclusive), igual a 2 si el valor de la suma cae en un tercer intervalo (por ejemplo, *x* es un número entero entre 10 y 21, inclusive), e igual a 3 si el valor de la suma cae en un cuarto intervalo (por ejemplo, *x* > 21). El elemento sintáctico se binariza usando un código Rice de K-ésimo orden, donde K es igual al parámetro de Rice determinado. También se pueden usar intervalos distintos de los proporcionados en el ejemplo anterior. Los intervalos pueden no superponerse.

[0099] Los procedimientos actuales de derivación de parámetro de Rice tienen al menos los siguientes inconvenientes. En primer lugar, en el diseño en HEVC, solo se toma en consideración un nivel de coeficiente codificado previo, que puede ser subóptimo. Además, dentro de un CG, el parámetro de Rice se mantiene sin cambios o sube, en el que en un área local, se pueden observar niveles de coeficiente más pequeños que prefieren parámetros de Rice más pequeños para bins menos codificados. Por ejemplo, dado que K se establece en 0 al comienzo de cada CG, y siempre que K sea menor que 4, K se incrementa cuando el absCoeffLevel de un coeficiente es mayor que $3 * 2^K$ o de lo contrario, se mantiene en el mismo valor. Sin embargo, en este ejemplo, el diseño de HEVC no permite que se disminuya K, incluso si hacerlo produciría una mejor compresión.

[0100] En segundo lugar, aunque el procedimiento de inicialización empleado en las extensiones de rango de HEVC (*véase* JCTVC-P0199) es muy beneficioso para la codificación sin pérdidas o con una gran profundidad de bits, todavía tiene algunos inconvenientes. Por ejemplo, dentro de un CG, el parámetro de Rice permanece sin cambios, se pueden observar niveles de coeficiente más pequeños que prefieren parámetros de Rice más pequeños para bins menos codificados. Por ejemplo, en la técnica en Extensiones de rango de HEVC, el parámetro de Rice se determina en base a un valor statCoeff, que se actualiza como máximo una vez por subbloque de 4 × 4. En la codificación sin pérdidas, se puede omitir la cuantificación. En la codificación con pérdida, se puede realizar la cuantificación y, como resultado, se pierde la información de los datos de vídeo originales.

[0101] En tercer lugar, el diseño en JCTVC-H0228 es más eficiente para la codificación con pérdida teniendo en cuenta a varios vecinos. Sin embargo, la correlación entre diferentes TU no se utiliza, como lo hacen las extensiones de rango de HEVC. Además, como se señala en la ecuación (8), anteriormente, el parámetro máximo de Rice producido en JCTVC-H0228 es igual a 3. Los errores de predicción (es decir, los valores de muestras residuales) pueden ser bastante grandes para la codificación sin pérdida o la codificación de alta profundidad de bits. En este caso, el parámetro de Rice máximo establecido en JCTVC-H0228 igual a 3 podría no ser lo suficientemente eficaz. Adicionalmente, se desconoce cómo establecer el parámetro de Rice basado en *sum_absolute_levelMinus1*.

[0102] Para resolver algunos o todos los problemas mencionados anteriormente, se proponen las siguientes técnicas para la derivación de parámetros de Rice y el modelado de contexto de coeficiente de transformada potencialmente más eficaz. Los siguientes procedimientos detallados pueden aplicarse individualmente. De forma alternativa, se puede aplicar cualquier combinación de los mismos. Las técnicas descritas en esta divulgación también se pueden usar conjuntamente con las técnicas propuestas en la Solicitud de Patente Provisional de EE. UU. 62/168.571, presentada el 29 de mayo de 2015, la Solicitud de Patente de EE. UU. 15/166.153, presentada el 26 de mayo de 2016, y la solicitud del PCT PCT/US2016/034828, presentada el 27 de mayo de 2016.

[0103] De acuerdo con una técnica ejemplar de esta divulgación, los procedimientos de derivación de parámetro basados en estadísticas y en plantillas se pueden usar por separado para codificar/descodificar niveles de coeficientes en una TU. Por ejemplo, de acuerdo con esta técnica, un codificador de vídeo puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficientes de una única TU de una CU actual de una imagen. En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas. Además, en este ejemplo, el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas.

[0104] Más específicamente, en un ejemplo, el procedimiento basado en estadísticas se aplica al primer elemento sintáctico *coeff_abs_level_remaining* en orden de descodificación/análisis en cada TU en el lado del descodificador, mientras que para otros elementos sintácticos *coeff_abs_level_remaining*, se aplica el procedimiento de derivación basado en plantillas. Por ejemplo, un codificador de vídeo puede actualizar el valor de statCoeff como se describe anteriormente para subbloques de 4 × 4 en base al valor del primer elemento sintáctico *coeff_abs_level_remaining* codificado de los subbloques. Además, en este ejemplo, el codificador de vídeo puede inicializar un parámetro de Rice basado en el valor de statCoeff. El codificador de vídeo puede usar a continuación este parámetro de Rice para codificar el primer elemento sintáctico *coeff_abs_level_remaining* de una TU. Adicionalmente, en este ejemplo, el codificador de vídeo puede realizar el procedimiento de derivación basado en plantillas como se describe anteriormente para cada elemento sintáctico *coeff_abs_level_remaining* de la TU que no sea el primer elemento sintáctico *coeff_abs_level_remaining* de la TU.

[0105] De forma alternativa, en algunos ejemplos, el procedimiento basado en estadísticas se aplica a los primeros elementos sintácticos *coeff_abs_level_remaining* en orden de codificación/descodificación en cada TU, mientras que para otros elementos sintácticos *coeff_abs_level_remaining*, se aplica el procedimiento de derivación basado en plantillas. Por tanto, en algunos ejemplos, el codificador de vídeo 20 puede señalizar y el descodificador de vídeo 30 puede obtener una serie de elementos sintácticos (por ejemplo, elementos sintácticos *coeff_abs_level_remaining*), indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU. En dichos ejemplos, como parte de un codificador de vídeo (por ejemplo, codificador de vídeo 20 o descodificador de vídeo 30) que usa el primer procedimiento de derivación de parámetro de Rice (por ejemplo, el procedimiento de derivación basado en estadísticas) y el segundo procedimiento de derivación de parámetro de Rice (por ejemplo, el procedimiento de derivación basado en plantillas) para niveles de coeficiente de descodificación de la TU, el codificador de vídeo puede aplicar el primer procedimiento de derivación de parámetro de Rice a un elemento sintáctico que aparece primero en la TU en un orden de descodificación o análisis, donde el elemento sintáctico está en la serie de elementos sintácticos. Adicionalmente, en este ejemplo, el codificador de vídeo puede aplicar el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos.

[0106] De forma alternativa, en algunos ejemplos, el procedimiento de derivación basado en estadísticas se aplica a uno o más elementos sintácticos *coeff_abs_level_remaining* para coeficientes localizados en posiciones relativas específicas en cada TU, mientras que para otros elementos sintácticos *coeff_abs_level_remaining*, se aplica el procedimiento de derivación basado en plantillas. Por ejemplo, el codificador de vídeo puede usar el procedimiento de derivación basado en estadísticas para determinar los parámetros de Rice para codificar los elementos sintácticos *coeff_abs_level_remaining* para el primer, el 16.º, ... y el 32.º coeficientes de una TU, y usar el procedimiento de derivación basado en plantillas para derivar parámetros de Rice para codificar cada uno de los otros elementos sintácticos *coeff_abs_level_remaining* de la TU.

[0107] De forma alternativa, en algunos ejemplos, el procedimiento basado en estadísticas se aplica al primer elemento sintáctico *coeff_abs_level_remaining* en orden de descodificación/análisis en cada Grupo de Coeficiente (CG) en el lado del descodificador, mientras que para otros elementos sintácticos *coeff_abs_level_remaining* en el CG, se aplica el procedimiento de derivación basado en plantillas. Por ejemplo, un codificador de vídeo (por ejemplo, codificador de vídeo 20 o descodificador de vídeo 30) puede, para cada subbloque de 4×4 de una TU, usar el procedimiento de derivación basado en estadísticas para determinar los parámetros de Rice para codificar un elemento sintáctico *coeff_abs_level_remaining* para un coeficiente que ocurre primero en un orden de análisis/descodificación usado en la descodificación del subbloque respectivo. En este ejemplo, el codificador de vídeo usa el procedimiento de derivación basado en plantillas para derivar un parámetro de Rice para codificar un elemento sintáctico *coeff_abs_level_remaining* para cada uno de los otros coeficientes de la TU.

[0108] De forma alternativa, en algunos ejemplos, el procedimiento basado en estadísticas se aplica al primero de los pocos elementos sintácticos *coeff_abs_level_remaining* en orden de descodificación/análisis en cada CG en el lado del descodificador, mientras que para otro *coeff_abs_level_remaining* en el CG, se aplica el procedimiento de derivación basado en plantillas. Por ejemplo, un codificador de vídeo (por ejemplo, codificador de vídeo 20 o descodificador de vídeo 30) puede, para cada subbloque de 4×4 de una TU, usar el procedimiento de derivación basado en estadísticas para determinar los parámetros de Rice para codificar un elemento sintáctico *coeff_abs_level_remaining* para un coeficiente que ocurre primero, por ejemplo, en las primeras dos o tres posiciones, en un orden de análisis/descodificación usado en la descodificación del subbloque respectivo. En este ejemplo, el codificador de vídeo usa el procedimiento de derivación basado en plantillas para derivar un parámetro de Rice para

codificar un elemento sintáctico *coeff_abs_level_remaining* para cada uno de los otros coeficientes de la TU.

[0109] De acuerdo con una técnica ejemplar de esta divulgación, los procedimientos de derivación de parámetros de Rice basados en estadísticas y en plantillas se usan conjuntamente para determinar un parámetro de Rice para codificar/descodificar un nivel de coeficiente. De esta manera, un codificador de vídeo puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice conjuntamente para codificar un solo nivel de coeficiente de una unidad de transformada, donde el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas, y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas.

[0110] Por tanto, en este ejemplo, un codificador de vídeo puede usar una función para determinar un valor de parámetro de Rice para descodificar un nivel de coeficiente de un coeficiente actual de una TU. En este ejemplo, las entradas de la función incluyen un valor registrado y un valor de parámetro de Rice derivado actual. El valor registrado puede basarse en un valor de parámetro de Rice usado para descodificar un coeficiente previo al coeficiente actual. En este ejemplo, el codificador de vídeo puede derivar el valor actual del parámetro de Rice derivado usando el procedimiento de derivación basado en plantillas. Además, en este ejemplo, la función puede ser una función máxima.

[0111] Por ejemplo, en un ejemplo, el parámetro de Rice usado para codificar/descodificar un elemento sintáctico *coeff_abs_level_remaining* actual se determina usando una función, con un valor registrado basado en el valor del parámetro de Rice usado para codificar/descodificar el coeficiente anterior y el valor de parámetro de Rice derivado actual usando el procedimiento basado en plantillas como entradas. A continuación, se describen ejemplos de determinación del valor registrado.

[0112] En un ejemplo donde el parámetro de Rice usado para programar códigos (es decir, codificar o descodificar) un elemento sintáctico *coeff_abs_level_remaining* actual se determina usando una función, la función es una función máxima, por ejemplo, el parámetro de Rice usado para codificar/descodificar el elemento sintáctico *coeff_abs_level_remaining* actual es el valor máximo entre un valor de parámetro de Rice registrado y un valor derivado actual usando el procedimiento basado en plantillas. En un ejemplo, el parámetro de Rice realmente usado para codificar/descodificar un elemento sintáctico *coeff_abs_level_remaining* se establece como el valor registrado para codificar/descodificar el siguiente elemento sintáctico *coeff_abs_level_remaining*. De forma alternativa, en algunos ejemplos, después de codificar/descodificar un elemento sintáctico *coeff_abs_level_remaining*, el valor del parámetro de Rice registrado para codificar/descodificar el siguiente elemento sintáctico *coeff_abs_level_remaining* se establece igual al parámetro de Rice usado realmente menos una variable *Diff.* En dicho ejemplo, la variable *Diff* se establece igual a 1. Por ejemplo, un codificador de vídeo puede establecer el valor registrado igual al valor del parámetro de Rice determinado para descodificar el nivel de coeficiente del coeficiente actual, menos 1. De forma alternativa, la variable *Diff* depende de la profundidad de bits interna codificada y/o de entrada y/o del modo de codificación. En algunos ejemplos, el codificador de vídeo puede determinar si el valor registrado es menor que 0. En base a que el valor registrado sea menor que 0, el codificador de vídeo puede restablecer el valor registrado a 0.

[0113] Como se describe anteriormente, se puede usar un valor registrado como parámetro de Rice para codificar un elemento sintáctico *coeff_abs_level_remaining*. En diferentes ejemplos, un codificador de vídeo puede determinar el valor registrado de diferentes maneras. Por ejemplo, en un ejemplo, al comienzo de la descodificación de una unidad de transformada o CG, el valor del parámetro de Rice registrado se establece igual a 0. De forma alternativa, en algunos ejemplos, el valor del parámetro de Rice registrado se establece igual al derivado basado en el procedimiento basado en estadísticas. En otras palabras, un codificador de vídeo puede establecer el valor del parámetro de Rice registrado usando el procedimiento basado en estadísticas. De forma alternativa, en algunos ejemplos, al descodificar el primer elemento sintáctico *coeff_abs_level_remaining* en una TU o CG, el parámetro de Rice derivado basado en el procedimiento basado en estadísticas se usa directamente sin considerar el valor anterior del parámetro de Rice. En otras palabras, al descodificar el elemento sintáctico *coeff_abs_level_remaining* que aparece primero en una TU (o, en algunos ejemplos, adicionalmente primero en cada CG de la TU), el codificador de vídeo usa el parámetro de Rice derivado directamente en base al procedimiento basado en estadísticas, sin considerar cualquier valor previo del parámetro de Rice. De forma alternativa, en algunos ejemplos, el 'procedimiento basado en plantillas' puede ser reemplazado por el 'procedimiento basado en retrospectiva' en los ejemplos anteriores.

[0114] De acuerdo con técnicas de ejemplo particulares de esta divulgación, en el procedimiento basado en plantillas, se propone una función genérica para derivar el parámetro K de Rice de *sum_absolute_levelMinus1.* Como se indica anteriormente, en el procedimiento basado en plantillas, el valor *sum_absolute_levelMinus1* puede indicar una suma de valores, siendo cada valor respectivo un valor absoluto de un nivel distinto de cero menos 1 de un coeficiente respectivo cubierto por una plantilla local para un coeficiente actual. Por tanto, de acuerdo con algunos de dichos ejemplos de esta divulgación, un codificador de vídeo puede determinar un valor (por ejemplo, *sum_absolute_levelMinus1*) igual a una suma absoluta de cada nivel de una pluralidad de niveles menos 1, donde cada nivel de coeficiente respectivo de la pluralidad de niveles está en una región definida por una plantilla. En este ejemplo, el codificador de vídeo puede determinar un parámetro de Rice basado en un valor y generar, basado en el parámetro de Rice, un valor descodificado para un coeficiente actual de la TU.

[0115] Por ejemplo, en un ejemplo donde un codificador de vídeo usa una función genérica para derivar el parámetro

de Rice de *sum_absolute_levelMinus1,* K se define como el número entero mínimo que satisface (1 << (K+3)) > (*sum_absolute_levelMinus1* + M), en la que M es un número entero, por ejemplo, 4. De forma alternativa, en algunos ejemplos, '>' se remplaza por '>='. Por ejemplo, que M sea igual a 4 y que *sum_absolute_levelMinus1* sea igual a 5, por tanto (*sum_absolute_levelminus1* + M) es igual a 9. En este ejemplo, si K es igual a 0, (1 << (K+3)) es igual a 8 y si K es igual a 1, (1 << (K+3)) es igual a 16. Por lo tanto, en este ejemplo, dado que K = 1 es el valor entero mínimo que genera (1 << (K+3)) > (*sum_absolute_levelminus1* + M), el codificador de vídeo establece K igual a 1.

[0116] En un ejemplo, K se define como el número entero mínimo que satisface (1 << (K+3)) > (*sum_absolute_levelminus1* + M + (1<<K)), en el que M es un número entero, por ejemplo, 4. De forma alternativa, en algunos ejemplos, '>' se remplaza por '>='. Por ejemplo, sea M igual a 4 y sea *sum_absolute_levelminus1* igual a 5. En este ejemplo, si K es igual a 0, (1 << (K+3)) es igual a 8 y (*sum_absolute_levelminus1* + M + (1<<K)) es igual a 10. Además, en este ejemplo, si K es igual a 1, (1 << (K+3)) es igual a 16 y (*sum_absolute_levelminus1* + M + (1<<K)) es igual a 11. Por lo tanto, en este ejemplo, dado que K = 1 es el valor entero mínimo que genera (1 << (K+3)) > (*sum_absolute_levelminus1* + M + (1<<K)), el codificador de vídeo establece K igual a 1.

[0117] De forma alternativa, en algunos ejemplos donde un codificador de vídeo usa una función genérica para derivar el parámetro de Rice de *sum_absolute_levelminus1*, la *K* derivada puede estar limitada a un umbral. Por ejemplo, el umbral puede ser 9, 10 u otro valor. En un ejemplo, el umbral puede predefinirse o señalizarse para una secuencia (por ejemplo, una secuencia de vídeo codificada) y/o para una imagen y/o para un fragmento. Por ejemplo, en diversos ejemplos, el umbral puede definirse en un conjunto de parámetros de secuencia, un conjunto de parámetros de imagen y/o una cabecera de fragmento. En otro ejemplo, el umbral depende del modo de codificación (codificación sin pérdida o codificación con pérdida) y/o profundidad de bits interna de entrada/codificada. Por ejemplo, el umbral puede ser mayor en la codificación sin pérdida que en la codificación con pérdida. De forma similar, el umbral puede ser mayor cuando se codifican coeficientes con mayor profundidad de bits interna.

[0118] De acuerdo con una técnica de ejemplo de esta divulgación, en el procedimiento de derivación de parámetro basado en plantillas, se definen y utilizan múltiples plantillas (es decir, localizaciones relativas de vecinos para calcular *sum_absolute_levelminus1*). En un ejemplo donde se definen y utilizan múltiples plantillas, la selección de la plantilla se basa en un patrón de escaneo. Por tanto, de acuerdo con este ejemplo, un codificador de vídeo puede seleccionar, entre una pluralidad de plantillas, una plantilla para usar en la derivación de un parámetro de Rice, indicando cada plantilla respectiva de la pluralidad de plantillas diferentes localizaciones de coeficientes vecinos en relación con un coeficiente actual. Además, el codificador de vídeo puede usar el parámetro de Rice para descodificar una palabra de código para un valor que indica un nivel absoluto de un valor restante para el coeficiente actual.

[0119] En algunos ejemplos donde se definen y utilizan múltiples plantillas, el patrón de escaneo puede ser el patrón usado para escanear coeficientes de una TU, tal como el patrón que se muestra en el ejemplo de la FIG. 3. Por ejemplo, un codificador de vídeo puede seleccionar una primera plantilla (por ejemplo, la plantilla de la FIG. 4) si se usa un primer patrón de escaneo (por ejemplo, el patrón de escaneo de la FIG. 3), pero puede seleccionar una segunda plantilla diferente si se usa un segundo patrón de escaneo diferente. Por ejemplo, si el segundo patrón de escaneo es el reverso del patrón de escaneo de la FIG. 3, la segunda plantilla puede ser similar a la plantilla de la FIG. 4, pero invertida vertical y horizontalmente.

[0120] En algunos ejemplos donde se definen y utilizan múltiples plantillas, la selección de la plantilla se basa en modos de intrapredicción. Por ejemplo, un codificador de vídeo puede seleccionar una primera plantilla para su uso en la determinación de los parámetros de Rice para codificar elementos sintácticos *coeff_abs_level_remaining* de coeficientes de TU de una CU si un modo de intrapredicción usado para intrapredecir una PU de la CU es horizontal o casi horizontal y puede seleccionar una segunda plantilla diferente si el modo de intrapredicción es vertical o casi vertical.

[0121] En algunos ejemplos donde se definen y utilizan múltiples plantillas, la selección de la plantilla se basa en modos de codificación (por ejemplo, modos intra o intercodificados) y/o matrices de transformada. Por ejemplo, un codificador de vídeo puede seleccionar una primera plantilla para su uso en la determinación de los parámetros de Rice para codificar elementos sintácticos de *coeff_abs_level_remaining* de coeficientes de TU de una CU si la CU está codificada usando intrapredicción y puede seleccionar una segunda plantilla diferente si la CU está codificada usando interpredicción.

[0122] En algunos ejemplos donde se definen y utilizan múltiples plantillas, la selección de la plantilla se basa en un parámetro de cuantificación (QP) de la TU actual. El QP puede ser una variable usada por un proceso de descodificación para ajustar a escala los niveles del coeficiente de transformada. Por ejemplo, un codificador de vídeo puede seleccionar una primera plantilla para su uso en la determinación de los parámetros de Rice para codificar elementos sintácticos de *coeff_abs_level_remaining* de coeficientes de TU de una CU si el QP es menor que un valor umbral y puede seleccionar una segunda plantilla diferente si el QP es mayor que el valor umbral.

[0123] En otro ejemplo, la selección de la plantilla podría basarse en la localización de un CG o la localización del nivel de coeficiente relativo a la TU y/o los tamaños de las unidades de transformada. Por ejemplo, un codificador de vídeo puede seleccionar una primera plantilla para su uso en la determinación de los parámetros de Rice para codificar

elementos sintácticos de *coeff_abs_level_remaining* de coeficientes de una fila inferior de CG de una TU de una CU y puede seleccionar una segunda plantilla diferente para cada otro CG de la TU.

[0124] Como se indica anteriormente, en el procedimiento de derivación basado en retrospectiva usado en HEVC, si el valor de *absCoeffLevel* para un nivel de coeficiente actual es mayor que $3*2^K$, un codificador de vídeo actualiza K para que sea igual al menor de K+1 y 4. Si el valor de *absCoeffLevel* del nivel del coeficiente actual no es mayor que $3*2^K$, el codificador de vídeo no actualiza K. De acuerdo con una técnica de ejemplo de esta divulgación, en el procedimiento de derivación basado en retrospectiva, en lugar de usar el valor absoluto (*absCoeffLevel*) para calcular el parámetro de Rice, se usa el valor de un elemento sintáctico *coeff_abs_level_remaining* a codificar (que es igual a (*absCoeffLevel* − *baseLevel*) como se define en la ecuación (1)). Por tanto, en este ejemplo, si el valor del elemento sintáctico *coeff_abs_level_remaining* para un nivel de coeficiente actual es mayor que $3*2^K$, un codificador de vídeo actualiza K para que sea igual al menor de K+1 y 4. Si el valor *absCoeffLevel* del nivel de coeficiente actual no es mayor que $3*2^K$, el codificador de vídeo no actualiza K. En este ejemplo, el codificador de vídeo puede usar a continuación el valor resultante de K como el parámetro de Rice para codificar el elemento sintáctico *coeff_abs_level_remaining* para el nivel de coeficiente actual.

[0125] Por ejemplo, un codificador de vídeo puede determinar un valor restante de nivel (por ejemplo, *coeff_abs_level_remaining*) para un coeficiente anterior, siendo el valor restante de nivel igual a una diferencia entre un primer valor y un segundo valor, el primer valor entre un valor absoluto del coeficiente anterior, siendo el segundo valor igual a una suma de un primer indicador (por ejemplo, *significant_flag),* un segundo indicador (por ejemplo, *coeff_abs_level_reater1_flag),* y un tercer indicador (por ejemplo, *coeff_abs_level_greater2_flag).* El primer indicador indica si el coeficiente anterior es distinto de cero, el segundo indicador indica si el coeficiente anterior es mayor que 1 y el tercer indicador indica si el coeficiente anterior es mayor que 2. En este ejemplo, el codificador de vídeo puede determinar, en base al valor restante de nivel para el coeficiente anterior, si se debe modificar un parámetro de Rice. Además, en este ejemplo, el codificador de vídeo puede usar el parámetro de Rice para descodificar una palabra de código para un nivel absoluto de un valor restante para un coeficiente actual.

[0126] Como se indica anteriormente, en el procedimiento de derivación basado en estadísticas usado en las extensiones de rango de HEVC, el valor *statCoeff* se incrementa si el nivel *absCoeffLevel* de un nivel de coeficiente actual es mayor o igual a (3*(1<<(*statCoeff* / 4))) y disminuye si ((2**absCoeffLevel*) < (1 << (*statCoeff* / 4))). De acuerdo con una técnica de ejemplo de esta divulgación, en el procedimiento de derivación basado en estadísticas, en lugar de usar el valor absoluto (*absCoeffLevel*) para actualizar la estadística *statCoeff,* se usa un elemento sintáctico *coeff abs level remaining* que se codificará (que es igual a (*absCoeffLevel-baseLevel*) como se define en la ecuación (1)). Por tanto, en este ejemplo, si el elemento sintáctico *coeff_abs_level_remaining* de un nivel de coeficiente actual es mayor o igual a (3*(1«(*statCoeff*/4))) y disminuye si ((2**coeff_abs_level_remaining*) < (1 << (*statCoeff*/4))). En este ejemplo, el codificador de vídeo puede establecer K igual al que sea menor de un parámetro máximo de Rice y (*statCoeff*/4). En este ejemplo, el codificador de vídeo puede usar K a continuación como parámetro de Rice para codificar el elemento sintáctico *coeff_abs_level_remaining* para el nivel de coeficiente actual.

[0127] Por ejemplo, de acuerdo con este ejemplo, un codificador de vídeo puede determinar un valor restante de nivel (por ejemplo, *coeff_abs_leve_remaining*) para un coeficiente anterior, siendo el valor restante de nivel igual a una diferencia entre un primer valor y un segundo valor, el primer valor entre un valor absoluto (por ejemplo, *absCoeffLevel)* del coeficiente anterior, el segundo valor (por ejemplo, *baseLevel*) igual a una suma de un primer indicador (por ejemplo, *significant_flag),* un segundo indicador (por ejemplo, *coeff_abs_level_greater1_flag),* y un tercer indicador (por ejemplo, *coeff_abs_level_greater2_flag*). El primer indicador indica si el coeficiente anterior es distinto de cero, el segundo valor indica si el coeficiente anterior es mayor que 1 y el tercer indicador indica si el coeficiente anterior es mayor que 2. En este ejemplo, el codificador de vídeo puede actualizar una estadística basada en el valor de nivel restante. Adicionalmente, en este ejemplo, el codificador de vídeo puede determinar, en base a la estadística, un parámetro de Rice. Además, en este ejemplo, el codificador de vídeo puede usar el parámetro de Rice para descodificar una palabra de código para un nivel absoluto de un valor restante para un coeficiente actual.

[0128] De acuerdo con una técnica de ejemplo de esta divulgación, en cualquiera de los tres procedimientos (es decir, procedimiento basado en plantillas, basado en retrospectiva y basado en estadísticas) mencionados anteriormente, el punto de conmutación para el código Rice y el código Rice más el código Exp-Golomb puede ser igual a (M << K), en el que M también depende de K, pero podría ser diferente para diferentes valores de K. En un ejemplo, para K igual a 0, 1, 2, M se establece en 6, 5, 6, respectivamente. En este ejemplo, para otros valores de K, M se establece en 3. En otro ejemplo, cuando K es menor que 3, M se establece en 6 y para otros valores de K, M se establece en 3.

[0129] Por ejemplo, en este ejemplo, un codificador de vídeo puede determinar un punto de conmutación como M << K, donde M es igual a 1 << K y K es un número de bits en la porción de resto de una palabra de código. En este ejemplo, el codificador de vídeo puede seleccionar, en base al punto de conmutación, un procedimiento de codificación de Rice o un procedimiento de codificación de Exp-Golomb. Adicionalmente, en este ejemplo, el codificador de vídeo puede usar el procedimiento de codificación seleccionado para descodificar una palabra de código para un nivel absoluto de un valor restante para un coeficiente actual.

[0130] De acuerdo con una técnica de ejemplo de esta divulgación, el procedimiento de derivación para un parámetro de Rice puede depender además de la profundidad de bits del vídeo de entrada o de la profundidad de bits del códec interno. En un ejemplo, cuando la profundidad de bits es igual a 8, se aplica el procedimiento basado en plantillas, y cuando la profundidad de bits es mayor que 8, por ejemplo, 10 o 12, se aplica el procedimiento basado en estadísticas.

[0131] Por tanto, de acuerdo con dicho ejemplo, un codificador de vídeo puede determinar un parámetro de Rice basado en una profundidad de bits de vídeo de entrada o basado en una profundidad de bits de códec interno. En este ejemplo, el codificador de vídeo puede usar el parámetro de Rice para descodificar una palabra de código para un nivel absoluto de un valor restante para un coeficiente actual.

[0132] De acuerdo con una técnica de ejemplo de esta divulgación, la codificación con pérdida o sin pérdida puede usar diferentes procedimientos para derivar el parámetro de Rice. Por ejemplo, un codificador de vídeo puede usar un procedimiento de derivación basado en plantillas si un bloque se codifica usando codificación con pérdida y puede usar un procedimiento de derivación basado en estadísticas si el bloque se codifica usando codificación sin pérdida. En otro ejemplo, un codificador de vídeo puede usar un procedimiento de derivación basado en estadísticas si un bloque se codifica usando codificación con pérdida y puede usar un procedimiento de derivación basado en plantillas si el bloque se codifica usando codificación sin pérdida.

[0133] Por tanto, de acuerdo con dicho ejemplo, un codificador de vídeo puede derivar un parámetro de Rice usando diferentes procedimientos dependiendo de si se usa codificación con pérdida o sin pérdida. Además, en este ejemplo, el codificador de vídeo puede usar el parámetro de Rice para descodificar una palabra de código para un nivel absoluto de un valor restante para un coeficiente actual.

[0134] De forma alternativa, en algunos ejemplos, se pueden aplicar diferentes procedimientos para derivar el parámetro de Rice de acuerdo con el valor del parámetro de cuantificación (QP) de la TU actual. Por ejemplo, un codificador de vídeo puede usar un procedimiento de derivación basado en plantillas si el valor de QP de la TU actual es mayor que un umbral y puede usar un procedimiento de derivación basado en estadísticas si el valor de QP de la TU actual es menor que el umbral. En otro ejemplo, un codificador de vídeo puede usar un procedimiento de derivación basado en estadísticas si el valor de QP de la CU actual es mayor que un umbral y puede usar un procedimiento de derivación basado en plantillas si el valor de QP de la CU actual es menor que el umbral.

[0135] Como se indica anteriormente, en algunos ejemplos, un codificador de vídeo usa códigos Rice para elementos sintácticos *coeff_abs_level_remaining* que tienen valores más pequeños y usa códigos Exp-Golomb para elementos sintácticos *coeff_abs_level_remaining* que tienen valores mayores. Tanto los códigos Rice como los códigos Exp-Golomb se determinan usando un parámetro K. De acuerdo con una técnica de ejemplo de esta divulgación, los procedimientos anteriores se pueden usar para determinar los órdenes tanto del código Rice como del código Exp-Golomb. Por ejemplo, un codificador de vídeo puede usar cualquiera de los ejemplos proporcionados anteriormente para determinar el valor K para su uso en la codificación de un elemento sintáctico *coeff_abs_level_remaining* para un nivel de coeficiente particular y usar el valor determinado de K para determinar un código Rice de K-ésimo orden o un código Exp-Golomb de K-ésimo orden para el elemento sintáctico *coeff_abs_level_remaining* para el nivel de coeficiente particular. De forma alternativa, en algunos ejemplos, los procedimientos descritos en esta divulgación solo se usan para determinar el orden del código Rice o del código Exp-Golomb. En otras palabras, en dichos ejemplos, un codificador de vídeo puede usar los ejemplos proporcionados en esta divulgación para determinar un valor del parámetro K solo para determinar los códigos Rice de K-ésimo orden; o el codificador de vídeo puede usar los ejemplos proporcionados en esta divulgación para determinar un valor del parámetro K solo para determinar los códigos Exp-Golomg de K-ésimo orden.

[0136] De acuerdo con una técnica de ejemplo de esta divulgación, de forma alternativa, en algunos ejemplos, los procedimientos anteriores pueden aplicarse a otros tipos de procedimientos de binarización para codificar los niveles de coeficientes que requieren la determinación de un orden, tal como el código EG de K-ésimo orden.

[0137] De acuerdo con una técnica de ejemplo de esta divulgación, de forma alternativa, los procedimientos anteriores también pueden aplicarse a otros elementos sintácticos para los que se aplica un procedimiento de binarización de K-ésimo orden, con o sin cambio de plantilla. En un ejemplo, la codificación de las diferencias del vector de movimiento puede usar los procedimientos anteriores. Un bloque intercodificado se codifica de acuerdo con un vector de movimiento que apunta a un bloque de muestras de referencia que forman el bloque predictivo, y los datos residuales indican la diferencia entre el bloque codificado y el bloque predictivo. En lugar de señalizar el vector de movimiento directamente, un codificador de vídeo puede identificar un conjunto de candidatos de vectores de movimiento, que pueden especificar parámetros de movimiento de un vecino espacial o temporal del bloque. El codificador de vídeo puede señalizar un índice de un candidato de vector de movimiento seleccionado en el conjunto de candidatos de vector de movimiento y señalizar una diferencia entre el vector de movimiento del candidato de vector de movimiento seleccionado y el vector de movimiento del bloque. La diferencia señalizada puede denominarse "diferencia de vector de movimiento" (MVD). Similar al enfoque en HEVC para niveles de coeficiente de señalización, una MVD puede señalizarse en HEVC usando, para cada una de una dimensión horizontal x y una dimensión vertical y, un elemento sintáctico abs_mvd_greater0_flag, un elemento sintáctico abs_mvd_greater1_flag, un elemento sintáctico abs_mvd_minus2 y un elemento sintáctico mvd_sign_flag. Como se indica en la subcláusula 7.4.9.9 de la

especificación de HEVC, abs_mvd_minus2[compIdx] plus 2 especifica el valor absoluto de una diferencia de componente del vector de movimiento. La tabla 9-38 de la especificación de HEVC indica que abs_mvd_minus2 se binariza usando un código Exp-Golomb de 1.er orden. Sin embargo, de acuerdo con un ejemplo de esta divulgación, el orden K de un código Rice o Exp-Golomb usado para binarizar un elemento sintáctico abs_mvd_minus2 (u otro elemento sintáctico para señalizar una MVD) se puede determinar usando cualquiera de los ejemplos proporcionados en otra parte de esta divulgación.

[0138] De acuerdo con un ejemplo de esta divulgación que usa un procedimiento de derivación de parámetro de Rice basado en plantillas, se define una función *sum_template (k)* para devolver el número de coeficientes en una plantilla cuyas magnitudes son mayores que k como:

$$sum\_template\ (k) = \sum \delta_j (x_i, k)$$

con $\delta_j(x,k) = \begin{cases} 1 & |x_i| > k \\ 0 & x_i = 0 \end{cases}$. Además, en este ejemplo, las funciones *f (x, y, n, t)* se definen para manejar la información de posición y $\delta_k(u,v)$ se define para manejar la información del componente como sigue:

$$f(x,y,n,t) = \begin{cases} n & x + y < t \\ 0 & x + y \geq t \end{cases}$$

$$\delta_k(u,v) = \begin{cases} u & v = 0 \\ 0 & v \neq 0 \end{cases}$$

Esta plantilla se muestra en la FIG. 4. El coeficiente de transformada actual se marca como 'X' y sus cinco vecinos espaciales se marcan como ’x_i’ (siendo i de 0 a 4). Si se cumple una de las siguientes dos condiciones, x_i se marca como no disponible y no se usa en el proceso de derivación de índice de contexto:

- La posición de x_i y el coeficiente de transformada actual X no se localizan en la misma unidad de transformada;
- La posición de x_i se localiza fuera del delimitador horizontal o vertical de la imagen
- El coeficiente de transformada x_i aún no se ha codificado/descodificado.

[0139] De acuerdo con algunas técnicas de esta divulgación, un codificador de vídeo puede realizar una selección de contexto para elementos sintácticos *significant_flag* (bin0) de niveles de coeficiente. Para realizar la selección de contexto, el codificador de vídeo puede determinar o calcular un índice de contexto que identifica el contexto seleccionado. De acuerdo con uno de dichos ejemplos, los cálculos del índice de contexto para un bin0 se definen como sigue:

- Para un bin0, el índice de contexto se deriva como:

$$c_0 = \min(sum\_template(0), 5) + f(x,y,6,2) + \delta_k(f(x,y,6,5), cIdx)$$

$$c_0 = c_0 + offset\ (cIdx, width)$$

en el que

$$offset(v, w) = \begin{cases} w == 4\ ?\ 0 : (w == 8\ ?\ NumberLumaCtxOneset : NumberLumaCtxOneset * 2) & v = 0 \\ NumberLumaCtxOneset * 3 & v \neq 0 \end{cases}$$

En las ecuaciones anteriores, antes de agregar el desplazamiento, c_0 es el índice de contexto para bin0 y c_0 puede variar de 0 a 17. Después de agregar el desplazamiento, c_0 puede variar de 0 a 53. Además, en las ecuaciones anteriores, la plantilla de suma de valor se determina como se describe $\delta_j(x,k) = \begin{cases} 1 & |x_i| > k \\ 0 & x_i = 0 \end{cases}$.

[0140] En algunos ejemplos, basados en el intervalo de c_0, un conjunto de contextos de luma incluye NumberLumaCtxOneset, es decir, 18 modelos de contexto. Diferentes tamaños de transformada (con la anchura de transformada indicada por ’*w*’) para codificar bin0 de luma pueden seleccionar su propio conjunto. Por ejemplo, el

codificador de vídeo puede usar un primer conjunto de contextos para codificar bin0 para muestras de luma en bloques de transformada de un primer tamaño, y puede usar un segundo conjunto diferente de contextos para codificar bin0 para muestras de luma en bloques de transformada de un segundo tamaño diferente.

[0141] Además, los contextos de croma y luma se pueden separar para mejorar aún más el rendimiento de la codificación. Por ejemplo, para entradas YCbCr, los tres componentes de color (es decir, Y, Cb y Cr) se pueden representar con un índice de componente v igual a 0, 1 y 2, respectivamente. Por tanto, en este ejemplo, un codificador de vídeo puede seleccionar un contexto de codificación para la codificación aritmética de un bin0 de una muestra de luma de un píxel y seleccionar un contexto de codificación diferente para la codificación aritmética de un bin0 de una muestra de croma del mismo píxel.

[0142] Además, en algunos ejemplos, un codificador de vídeo puede seleccionar un contexto de codificación aritmética de un bin1 (por ejemplo, un *coeff_abs_level_greater1_flag*) y un bin2 (por ejemplo, un *coeff_abs_level_greater2_flag*) de un nivel de coeficiente. Por ejemplo, para un bin1, el índice de contexto (c_1) se deriva como:

$$c_1 = min(sum\_template(1), 4) + N$$

$$c_1 = c_1 + \delta_k\left(f\left(x, y, 5, 3\right), cIdx\right) + \delta_k\left(f\left(x, y, 5, 10\right), cIdx\right)$$

Para un bin2, el índice de contexto (c_2) se deriva como:

$$c_2 = \min\left(sum\_template(2), 4\right) + N$$

$$c_2 = c_2 + \delta_k\left(f\left(x, y, 5, 3\right), cIdx\right) + \delta_k\left(f\left(x, y, 5, 10\right), cIdx\right)$$

En las ecuaciones anteriores, N es igual a 1 y *f(x, y, n, t)* se define como anteriormente (es decir, $f(x,y,n,t) = \begin{cases} n & x+y<t \\ 0 & x+y \geq t \end{cases}$). El primer bin1 o bin2 se codifica con el índice de contexto ci o c_2 igual a 0 y para otros bin1 y bin2, se codifican con un índice de contexto igual a ci o c_2, respectivamente, en el que c_1 y c_2 se definen anteriormente.

[0143] Como se indica anteriormente, un codificador de vídeo puede binarizar los bins restantes (por ejemplo, elementos sintácticos *coeff_abs_level_remaining*) para un nivel de coeficiente con un código Rice o, cuando está presente, un código Exp-Golomb. De acuerdo con este ejemplo, el codificador de vídeo puede usar tanto un procedimiento de derivación de parámetro de Rice basado en estadísticas como un procedimiento de derivación de parámetro de Rice basado en plantillas para codificar niveles de coeficiente de una TU de señal. Más específicamente, en este ejemplo, el codificador de vídeo usa tanto un procedimiento de derivación de parámetro de Rice basado en estadísticas como un procedimiento de derivación de parámetro de Rice basado en plantillas para codificar los bins restantes para niveles de coeficiente de una TU de señal. En este ejemplo, las siguientes reglas se aplican para decidir el orden K del parámetro de Rice del código Rice o del código Exp-Golomb.

- Antes de descodificar una unidad de transformada, se establece la variable *prevK* en 0
- Si es el primer valor que se descodifica, una variable K_{temp} se establece en $cRicePara_{Statistics}$. En otras palabras, si el elemento sintáctico *coeff_abs_level_remaining* que se codifica es el primer elemento sintáctico *coeff_abs_level_remaining* de una unidad de transformada que se va a codificar, una variable K_{temp} se establece en $cRicePara_{Statistics}$
- Si por el contrario (no es el primer valor que se descodifica), la variable K_{temp} se establece en $cRicePara_{Template}$
- Establecer K igual a max(prevK, K_{temp})
- Actualizar prev*K*: prev*K* = *K* - *M*

[0144] Por tanto, en este ejemplo, el codificador de vídeo puede aplicar un procedimiento de derivación del parámetro de Rice basado en estadísticas al primer elemento sintáctico *coeff_abs_level_remaining* de una TU y puede aplicar un procedimiento de derivación de parámetro de Rice basado en plantillas a cada elemento sintáctico *coeff_abs_level_remaining* subsiguiente de la TU. En un ejemplo, *M* se establece en 1. En otro ejemplo, *M* se establece en 0. Además, en algunos casos del ejemplo descrito en el párrafo anterior, el proceso de derivación de

cRicePara$_{Statistics}$ es el mismo que el proceso para determinar *cRiceParam,* como se describe en otra parte de esta divulgación. En algunos casos del ejemplo descrito en el párrafo anterior, el proceso de derivación de cRicePara$_{Template}$ sigue las siguientes etapas en orden:

- Se calcula *sum_absolute_levelMinus1* que sigue la forma usada en la ecuación (7) (es decir, $\delta_j(x) = \begin{cases} |x_i| - 1 & |x_i| > 0 \\ 0 & x_i = 0 \end{cases}$ *sum_absolute_levelMinus* 1 = $\sum\delta_j(x_i)$ con)
- Establecer *uiVal* igual a *sum_absolute_levelMinus1*
- Seleccionar K usando el siguiente pseudocódigo:

```
for ( iOrder = 0; iOrder < MAX_GR_ORDER_RESIDUAL; iOrder ++ )
{
  if( (1 << (iOrder + 3)) > (uiVal + 4))
  {
    break;
  }
}

   K = (iOrder == MAX_GR_ORDER_RESIDUAL ? (MAX_GR_ORDER_RESIDUAL
- 1): iOrder);
```

[0145] Por tanto, en este ejemplo, el codificador de vídeo establece K en el menor del orden máximo permitido o el valor no negativo más pequeño de *iOrder* donde (1 << (*iOrder* + 3)) > (*uiVal* + 4). En un ejemplo, MAX_GR_ORDER_RESIDUAL se establece en 10. En otro ejemplo, MAX _GR ORDER RESIDUAL se establece en 12.

[0146] De forma alternativa o además, la ecuación (7) se puede modificar como sigue:

$$sum\_absolute\_level = \sum |x_i|$$

$$sum\_absolute\_levelMinus1 = \sum \delta_j(x_i) \qquad (7)$$

con $\delta_j(x) = \begin{cases} ((|x_i| + offset) >> M) - 1 & |x_i| > 0 \\ 0 & x_i = 0 \end{cases}$ En un ejemplo, en la ecuación (7) anterior, M se establece igual a la profundidad de bits de entrada/interna menos 8. De forma alternativa, en la ecuación (7) anterior, M se establece en 0. De forma alternativa, el valor de M depende del modo de codificación con pérdida/sin pérdida, y/o de la profundidad de bits de entrada/interna y/o del modo de codificación intra/inter, y/o del valor señalado o el valor derivado de la información previamente codificada. En un ejemplo, el *desplazamiento* se establece en (1 << (M-1)). De forma alternativa, en algunos ejemplos, el desplazamiento se establece en 0. De forma alternativa, la función $\delta_j(x)$ se define como:

$$\delta_j(x) = \begin{cases} ((|x_i| + offset) >> M) & |x_i| > 0 \\ 0 & x_i = 0 \end{cases}$$

[0147] En algunos ejemplos, SP(K) es una función que devuelve un valor utilizado para determinar un punto de conmutación (es decir, un valor por encima del cual se usan códigos Exp-Golomb en lugar de códigos Rice). En un ejemplo, SP(0) = SP(2) = 6, SP(1) = 5, y para los otros valores de *K, SP(K)* es igual a 3. De forma alternativa, además, SP(K) se tiene en cuenta para las ecuaciones (5) y (6) en el procedimiento de derivación basado en estadísticas. En algunos ejemplos, la binarización de un elemento sintáctico *coeff_abs_level_remaining* se define como:

- Si el elemento sintáctico *coeff_abs_level_remaining* es menor que (SP(*K*) << *K),* se aplica el código Rice de orden *K*;

- Si por el contrario (*coeff_abs_level_remaining* es igual o mayor que (SP(*K*) << *K*)), se usan un prefijo y un sufijo como palabra de código, y el prefijo es un número SP(K) de '1' y el sufijo es un código Exp-Golomb de K-ésimo orden.

Tabla 1. Palabra de código de *coeff abs level remaining* igual a m

Valor m		***K***
	Prefijo	***Sufijo***
$0\sim2^K-1$	$X_1 \ldots X_{K-1}$	N/A
$2^K\sim 2*2^K-1$	$X_0\ X_1 \ldots X_{K-1}$	N/A
...		N/A
$(SP(K)-1)*2^K\sim SP(K)*2^K-1$	11..0 $X_0\ X_1 \ldots X_{K-1}$ nota: SP(K)-1 '1's	N/A
$SP(K)*2^K\sim SP(K)*2^K+2^K-1$	11...1*	$0\ X_0\ X_1 \ldots X_{K-1}$
$SP(K)*2^K+2^K\sim SP(K)*2^K+2^K+2^{K+1}-1$	11...1*	$10\ X_0\ X_1 \ldots X_{K-1}\ X_K$
$SP(K)*2^K+2^K+2^{K+1}\sim SP(K)*2^K+2^K+2^{K+1}+2^{K+2}-1$	11...1*	$110\ X_0\ X_1 \ldots X_{K-1}\ X_K\ X_{K+1}$
...	...	....
En la tabla anterior, * indica valores '1' de SP(K). Por ejemplo, si SP(K) es 6, * representa seis 1 consecutivos.		

[0148] La FIG. 5 es un diagrama de bloques que ilustra un codificador de vídeo ejemplar 20 que puede implementar las técnicas de esta divulgación. La FIG. 5 se proporciona con propósitos explicativos y no se debería considerar limitante de las técnicas ampliamente ejemplificadas y descritas en esta divulgación. Con propósitos explicativos, esta divulgación describe un codificador de vídeo 20 en el contexto de la codificación HEVC. Sin embargo, las técnicas de esta divulgación pueden ser aplicables a otras normas o procedimientos de codificación.

[0149] En el ejemplo de la FIG. 5, el codificador de vídeo 20 incluye una unidad de procesamiento de predicción 100, una memoria de datos de vídeo 101, una unidad de generación residual 102, una unidad de procesamiento de transformada 104, una unidad de cuantificación 106, una unidad de cuantificación inversa 108, una unidad de procesamiento de transformada inversa 110, una unidad de reconstrucción 112, una unidad de filtro 114, una memoria intermedia de imágenes descodificadas 116 y una unidad de codificación por entropía 118. La unidad de procesamiento de predicción 100 incluye una unidad de procesamiento de interpredicción 120 y una unidad de procesamiento de intrapredicción 126. La unidad de procesamiento de interpredicción 120 incluye una unidad de estimación de movimiento y una unidad de compensación de movimiento (no se muestran). En otros ejemplos, el codificador de vídeo 20 puede incluir más, menos o diferentes componentes funcionales.

[0150] La memoria de datos de vídeo 101 puede almacenar datos de vídeo a codificar mediante los componentes del codificador de vídeo 20. Los datos de vídeo almacenados en la memoria de datos de vídeo 101 se pueden obtener, por ejemplo, de la fuente de vídeo 18. La memoria intermedia de imágenes descodificadas 116 puede ser una memoria de imágenes de referencia que almacena datos de vídeo de referencia para su uso en la codificación de datos de vídeo mediante el codificador de vídeo 20, por ejemplo, en los modos de intracodificación o intercodificación. La memoria de datos de vídeo 101 y la memoria intermedia de imágenes descodificadas 116 pueden estar formadas por cualquiera de una variedad de dispositivos de memoria, tales como memoria dinámica de acceso aleatorio (DRAM), incluyendo DRAM síncrona (SDRAM), RAM magnetorresistiva (MRAM), RAM resistiva (RRAM) u otros tipos de dispositivos de memoria. El mismo dispositivo de memoria u otros dispositivos de memoria separados pueden proporcionar una memoria de datos de vídeo 101 y una memoria intermedia de imágenes descodificadas 116. En diversos ejemplos, la memoria de datos de vídeo 101 puede estar en un chip con otros componentes del codificador de vídeo 20, o fuera del chip en relación con esos componentes.

[0151] El codificador de vídeo 20 recibe datos de vídeo. El codificador de vídeo 20 puede codificar cada CTU en un fragmento de una imagen de los datos de vídeo. Cada una de las CTU puede estar asociada a bloques de árbol de codificación (CTB) de luma de igual tamaño y a CTB correspondientes de la imagen. Como parte de la codificación de una CTU, la unidad de procesamiento de predicción 100 puede realizar una división de árbol cuaternario para dividir los CTB de la CTU en bloques progresivamente más pequeños. Los bloques más pequeños pueden ser bloques de codificación de las CU. Por ejemplo, la unidad de procesamiento de predicción 100 puede dividir un CTB asociado a una CTU en cuatro subbloques de igual tamaño, dividir uno o más de los subbloques en cuatro subbloques de igual tamaño, y así sucesivamente.

[0152] El codificador de vídeo 20 puede codificar las CU de una CTU para generar representaciones codificadas de las CU (es decir, CU codificadas). Como parte de la codificación de una CU, la unidad de procesamiento de predicción 100 puede dividir los bloques de codificación asociados a la CU entre una o más PU de la CU. Por tanto, cada PU

puede estar asociada a un bloque de predicción de luma y a bloques correspondientes de predicción de croma. El codificador de vídeo 20 y el descodificador de vídeo 30 pueden admitir PU que tienen diversos tamaños. El tamaño de una CU se puede referir al tamaño del bloque de codificación de luma de la CU, y el tamaño de una PU se puede referir al tamaño de un bloque de predicción de luma de la PU. Suponiendo que el tamaño de una CU particular es de 2Nx2N, el codificador de vídeo 20 y el descodificador de vídeo 30 pueden admitir tamaños de PU de 2Nx2N o NxN para la intrapredicción, y tamaños de PU simétricos de 2Nx2N, 2NxN, Nx2N, NxN o similares para la interpredicción. El codificador de vídeo 20 y el descodificador de vídeo 30 también pueden admitir una partición asimétrica para tamaños de PU de 2NxnU, 2NxnD, nLx2N y nRx2N para la interpredicción.

[0153] La unidad de procesamiento de interpredicción 120 puede generar datos predictivos para una PU realizando una interpredicción en cada PU de una CU. Los datos predictivos para la PU pueden incluir bloques predictivos de la PU e información de movimiento para la PU. La unidad de procesamiento de interpredicción 120 puede realizar diferentes operaciones para una PU de una CU dependiendo de si la PU está en un fragmento I, un fragmento P o un fragmento B. En un fragmento I, todas las PU se intrapredicen. Por consiguiente, si la PU está en un fragmento I, la unidad de procesamiento de interpredicción 120 no realiza interpredicción en la PU. Por tanto, para bloques codificados en el modo I, el bloque predicho se forma usando predicción espacial a partir de bloques vecinos previamente codificados dentro de la misma trama. Si una PU está en un fragmento P, la unidad de procesamiento de interpredicción 120 puede usar interpredicción unidireccional para generar un bloque predictivo de la PU.

[0154] La unidad de procesamiento de intrapredicción 126 puede generar datos predictivos para una PU realizando una intrapredicción en la PU. Los datos predictivos para la PU pueden incluir bloques predictivos de la PU y diversos elementos sintácticos. La unidad de procesamiento de intrapredicción 126 puede realizar una intrapredicción en las PU en fragmentos I, fragmentos P y fragmentos B.

[0155] Para realizar una intrapredicción en una PU, la unidad de procesamiento de intrapredicción 126 puede usar múltiples modos de intrapredicción para generar múltiples conjuntos de datos predictivos para la PU. La unidad de procesamiento de intrapredicción 126 puede usar muestras de bloques de muestras de PU vecinas para generar un bloque predictivo para una PU. Las PU vecinas pueden estar arriba, arriba y a la derecha, arriba y a la izquierda, o a la izquierda de la PU, suponiendo un orden de codificación de izquierda a derecha y de arriba abajo, para las PU, CU y CTU. La unidad de procesamiento de intrapredicción 126 puede usar diversos números de modos de intrapredicción, por ejemplo, 33 modos de intrapredicción direccional. En algunos ejemplos, el número de modos de intrapredicción puede depender del tamaño de la región asociada a la PU.

[0156] La unidad de procesamiento de predicción 100 puede seleccionar los datos predictivos para las PU de una CU de entre los datos predictivos generados por la unidad de procesamiento de interpredicción 120 para las PU, o los datos predictivos generados por la unidad de procesamiento de intrapredicción 126 para las PU. En algunos ejemplos, la unidad de procesamiento de predicción 100 selecciona los datos predictivos para las PU de la CU en base a unas métricas de velocidad/distorsión de los conjuntos de datos predictivos. Los bloques predictivos de los datos predictivos seleccionados pueden denominarse en el presente documento bloques predictivos seleccionados.

[0157] La unidad de generación residual 102 puede generar, en base a los bloques de codificación (por ejemplo, bloques de codificación de luma, Cb y Cr) para una CU y los bloques predictivos seleccionados (por ejemplo, bloques predictivos de luma, Cb y Cr) para las PU de la CU, bloques residuales (por ejemplo, bloques residuales de luma, Cb y Cr) para la CU. Por ejemplo, la unidad de generación residual 102 puede generar los bloques residuales de la CU de modo que cada muestra en los bloques residuales tenga un valor igual a una diferencia entre una muestra en un bloque de codificación de la CU y una muestra correspondiente en un bloque predictivo seleccionado correspondiente de una PU de la CU.

[0158] La unidad de procesamiento de transformada 104 puede realizar una partición de árbol cuaternario para dividir los bloques residuales asociados a una CU en bloques de transformada asociados a las TU de la CU. Por tanto, una TU puede estar asociada a un bloque de transformada de luma y a dos bloques de transformada de croma. Los tamaños y las posiciones de los bloques de transformada de luma y croma de las TU de una CU pueden o no estar basados en los tamaños y las posiciones de bloques de predicción de las PU de la CU. Una estructura de árbol cuaternario conocida como "árbol cuaternario residual" (RQT) puede incluir nodos asociados a cada una de las regiones. Las TU de una CU pueden corresponder a nodos hoja del RQT.

[0159] La unidad de procesamiento de transformada 104 puede generar bloques de coeficientes de transformada para cada TU de una CU aplicando una o más transformadas a los bloques de transformada de la TU. La unidad de procesamiento de transformada 104 puede aplicar diversas transformadas a un bloque de transformada asociado a una TU. Por ejemplo, la unidad de procesamiento de transformada 104 puede aplicar una transformada discreta del coseno (DCT), una transformada direccional o una transformada conceptualmente similar a un bloque de transformada. En algunos ejemplos, la unidad de procesamiento de transformada 104 no aplica transformadas a un bloque de transformada. En dichos ejemplos, el bloque de transformada se puede tratar como un bloque de coeficientes de transformada.

[0160] La unidad de cuantificación 106 puede cuantificar los coeficientes de transformada en un bloque de

coeficientes. El proceso de cuantificación puede reducir la profundidad de bits asociada a algunos, o a la totalidad, de los coeficientes de transformada. Por ejemplo, un coeficiente de transformada de n bits puede redondearse hacia abajo hasta un coeficiente de transformada de m bits durante la cuantificación, donde *n* es mayor que *m*. La unidad de cuantificación 106 puede cuantificar un bloque de coeficientes asociado a una TU de una CU basándose en un valor de parámetro de cuantificación (QP) asociado a la CU. El codificador de vídeo 20 puede ajustar el grado de cuantificación aplicado a los bloques de coeficientes asociados a una CU, ajustando el valor QP asociado a la CU. La cuantificación puede introducir pérdida de información, por lo que los coeficientes de transformada cuantificados pueden tener una precisión menor que los originales.

[0161] La unidad de cuantificación inversa 108 y la unidad de procesamiento de transformada inversa 110 pueden aplicar una cuantificación inversa y transformadas inversas a un bloque de coeficientes, respectivamente, para reconstruir un bloque residual a partir del bloque de coeficientes. La unidad de reconstrucción 112 puede añadir el bloque residual reconstruido a las muestras correspondientes de uno o más bloques predictivos generados por la unidad de procesamiento de predicción 100 para generar un bloque de transformada reconstruido asociado a una TU. Reconstruyendo bloques de transformada para cada TU de una CU de esta manera, el codificador de vídeo 20 puede reconstruir los bloques de codificación de la CU.

[0162] La unidad de filtro 114 puede realizar una o más operaciones de desbloqueo para reducir los artefactos de los bloques en los bloques de codificación asociados a una CU. La memoria intermedia de imágenes descodificadas 116 puede almacenar los bloques de codificación reconstruidos después de que la unidad de filtro 114 realice las una o más operaciones de desbloqueo en los bloques de codificación reconstruidos. La unidad de procesamiento de interpredicción 120 puede usar una imagen de referencia que contiene los bloques de codificación reconstruidos para realizar una interpredicción en las PU de otras imágenes. Además, la unidad de procesamiento de intrapredicción 126 puede usar bloques de codificación reconstruidos de la memoria intermedia de imágenes descodificadas 116 para realizar una intrapredicción en otras PU de la misma imagen que la CU.

[0163] La unidad de codificación por entropía 118 puede recibir datos desde otros componentes funcionales del codificador de vídeo 20. Por ejemplo, la unidad de codificación por entropía 118 puede recibir bloques de coeficientes desde la unidad de cuantificación 106 y puede recibir elementos sintácticos desde la unidad de procesamiento de predicción 100. La unidad de codificación por entropía 118 puede realizar una o más operaciones de codificación por entropía en los datos para generar datos codificados por entropía. Por ejemplo, la unidad de codificación por entropía 118 puede realizar una operación de CABAC, una operación de codificación de longitud variable adaptativa al contexto (CAVLC), una operación de codificación de longitud variable a variable (V2V), una operación de codificación aritmética binaria adaptativa al contexto basada en la sintaxis (SBAC), una operación de codificación por entropía por división de intervalos de probabilidad (PIPE), una operación de codificación exponencial-Golomb u otro tipo de operación de codificación por entropía en los datos. El codificador de vídeo 20 puede emitir un flujo de bits que incluye datos codificados por entropía, generados por la unidad de codificación por entropía 118. Por ejemplo, el flujo de bits puede incluir datos que representan un RQT para una CU.

[0164] La unidad de codificación por entropía 118 puede configurarse para realizar técnicas propuestas en esta divulgación. Por ejemplo, la unidad de codificación por entropía 118 puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para codificar niveles de coeficientes de una única TU de una CU actual de una imagen. En este ejemplo, el primer procedimiento de derivación de parámetro de Rice puede ser un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice puede ser un procedimiento de derivación basado en plantillas. En este ejemplo, como parte de la codificación de un nivel de coeficiente usando tanto el primer como el segundo procedimiento de derivación de parámetro de Rice, la unidad de codificación por entropía 118 puede binarizar un elemento sintáctico asociado con el nivel de coeficiente (por ejemplo, un elemento sintáctico *coeff_abs_level remaining*) como una palabra de código de Rice o Exp-Golomb de K-ésimo orden, donde K se determina usando el primer o el segundo procedimiento de derivación de parámetro de Rice. Además, en este ejemplo, la unidad de codificación por entropía 118 puede realizar codificación aritmética en la palabra de código.

[0165] La FIG. 6 es un diagrama de bloques que ilustra un descodificador de vídeo ejemplar 30 que está configurado para implementar las técnicas de esta divulgación. La FIG. 6 se proporciona con propósitos explicativos y no se limita a las técnicas ampliamente ejemplificadas y descritas en esta divulgación. Con propósitos explicativos, esta divulgación describe un descodificador de vídeo 30 en el contexto de la codificación HEVC. Sin embargo, las técnicas de esta divulgación pueden ser aplicables a otras normas o procedimientos de codificación.

[0166] En el ejemplo de la FIG. 6, el descodificador de vídeo 30 incluye una unidad de descodificación por entropía 150, una memoria de datos de vídeo 151, una unidad de procesamiento de predicción 152, una unidad de cuantificación inversa 154, una unidad de procesamiento de transformada inversa 156, una unidad de reconstrucción 158, una unidad de filtro 160 y una memoria intermedia de imágenes descodificadas 162. La unidad de procesamiento de predicción 152 incluye una unidad de compensación de movimiento 164 y una unidad de procesamiento de intrapredicción 166. En otros ejemplos, el descodificador de vídeo 30 puede incluir más, menos o diferentes componentes funcionales.

[0167] La memoria de datos de vídeo 151 puede almacenar datos de vídeo, tales como un flujo de bits de vídeo codificado que los componentes del descodificador de vídeo 30 van a descodificar. Los datos de vídeo almacenados en memoria de datos de vídeo 151 se pueden obtener, por ejemplo, a partir del canal 16, por ejemplo, desde una fuente de vídeo local, tal como una cámara, por medio de una comunicación alámbrica o inalámbrica de datos de vídeo, o accediendo a medios físicos de almacenamiento de datos. La memoria de datos de vídeo 151 puede formar una memoria intermedia de imágenes codificadas (CPB) que almacena datos de vídeo codificados a partir de un flujo de bits de vídeo codificado. La memoria intermedia de imágenes descodificadas 162 puede ser una memoria de imágenes de referencia que almacena datos de vídeo de referencia para su uso en la descodificación de datos de vídeo mediante el descodificador de vídeo 30, por ejemplo, en los modos de intracodificación o intercodificación. La memoria de datos de vídeo 151 y la memoria intermedia de imágenes descodificadas 162 pueden estar formadas por cualquiera de una variedad de dispositivos de memoria, tales como memoria dinámica de acceso aleatorio (DRAM), incluyendo DRAM síncrona (SDRAM), RAM magnetorresistiva (MRAM), RAM resistiva (RRAM) u otros tipos de dispositivos de memoria. El mismo dispositivo de memoria u otros dispositivos de memoria separados pueden proporcionar una memoria de datos de vídeo 151 y una memoria intermedia de imágenes descodificadas 162. En diversos ejemplos, la memoria de datos de vídeo 151 puede estar en un chip con otros componentes del descodificador de vídeo 30, o fuera del chip con respecto a esos componentes.

[0168] La memoria de datos de vídeo 151 recibe y almacena datos de vídeo codificados (por ejemplo, unidades de NAL) de un flujo de bits. La unidad de descodificación por entropía 150 puede recibir datos de vídeo codificados (por ejemplo, unidades de NAL) desde la CPB y analizar las unidades de NAL para obtener elementos sintácticos. La unidad de descodificación por entropía 150 puede descodificar por entropía elementos sintácticos codificados por entropía en las unidades de NAL. La unidad de procesamiento de predicción 152, la unidad de cuantificación inversa 154, la unidad de procesamiento de transformada inversa 156, la unidad de reconstrucción 158 y la unidad de filtro 160 pueden generar datos de vídeo descodificados en base a los elementos sintácticos extraídos del flujo de bits. La unidad de descodificación por entropía 150 puede realizar un proceso en general recíproco al de la unidad de codificación por entropía 118.

[0169] Como parte de la descodificación del flujo de bits, la unidad de descodificación por entropía 150 puede extraer y descodificar por entropía elementos sintácticos de las unidades de NAL de fragmentos codificados. Cada uno de los fragmentos codificados puede incluir una cabecera de fragmento y datos de fragmento. La cabecera de fragmento puede contener elementos sintácticos pertenecientes a un fragmento. La unidad de codificación por entropía 118 puede configurarse para realizar técnicas propuestas en esta divulgación.

[0170] La unidad de descodificación por entropía 150 puede realizar al menos partes de diversos ejemplos de esta divulgación. Por ejemplo, la unidad de descodificación por entropía 150 puede desbinarizar elementos sintácticos binarizados, tales como elementos sintácticos *coeff_abs_level remaining*. Como parte de la desbinarización de elementos sintácticos binarizados, la unidad de descodificación por entropía 150 puede determinar parámetros de Rice de la manera descrita en los ejemplos proporcionados en otra parte de esta divulgación. Por ejemplo, de acuerdo con un ejemplo de esta divulgación, la unidad de descodificación por entropía 150 puede usar un primer procedimiento de derivación de parámetros de Rice y un segundo procedimiento de derivación de parámetros de Rice para descodificar niveles de coeficientes de una única TU de una CU actual de una imagen, en el que el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas, y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas.

[0171] En este ejemplo, como parte de la descodificación de un nivel de coeficiente usando tanto el primer como el segundo procedimiento de derivación de parámetro de Rice, la unidad de descodificación por entropía 150 puede realizar descodificación aritmética para recuperar una palabra de código para un elemento sintáctico asociado con el nivel de coeficiente (por ejemplo, un elemento sintáctico *coeff abs_level remaining*). En este ejemplo, la unidad de descodificación por entropía 150 puede interpretar la palabra de código como una palabra de código de Rice o Exp-Golomb de K-ésimo orden, donde K se determina usando el primer o segundo procedimiento de derivación de parámetro de Rice. Además, en este ejemplo, la unidad de descodificación por entropía 150 puede desbinarizar la palabra de código de Rice o Exp-Golomb convirtiéndola nuevamente en un número no codificado.

[0172] Además de obtener elementos sintácticos del flujo de bits, el descodificador de vídeo 30 puede realizar una operación de reconstrucción en una CU no dividida. Para realizar la operación de reconstrucción en una CU, el descodificador de vídeo 30 puede realizar una operación de reconstrucción en cada TU de la CU. Realizando la operación de reconstrucción para cada TU de la CU, el descodificador de vídeo 30 puede reconstruir bloques residuales de la CU.

[0173] Como parte de realizar una operación de reconstrucción en una TU de una CU, la unidad de cuantificación inversa 154 puede realizar la cuantificación inversa, es decir, descuantificar, los bloques de coeficientes asociados a la TU. La unidad de cuantificación inversa 154 puede usar un valor QP asociado a la CU de la TU para determinar un grado de cuantificación y, asimismo, un grado de cuantificación inversa para que la unidad de cuantificación inversa 154 lo aplique. Es decir, la relación de compresión, es decir, la relación entre el número de bits usados para representar la secuencia original y la comprimida, puede controlarse ajustando el valor del QP usado al cuantificar los coeficientes de transformada. La relación de compresión también puede depender del procedimiento de codificación por entropía

empleado.

[0174] Después de que la unidad de cuantificación inversa 154 haya realizado la cuantificación inversa de un bloque de coeficientes, la unidad de procesamiento de transformada inversa 156 puede aplicar una o más transformadas inversas al bloque de coeficientes con objeto de generar un bloque residual asociado a la TU. Por ejemplo, la unidad de procesamiento de transformada inversa 156 puede aplicar una DCT inversa, una transformada entera inversa, una transformada inversa de Karhunen-Loeve (KLT), una transformada de rotación inversa, una transformada direccional inversa u otra transformada inversa al bloque de coeficientes.

[0175] Si se codifica una PU usando intrapredicción, la unidad de procesamiento de intrapredicción 166 puede realizar intrapredicción para generar bloques predictivos de la PU. La unidad de procesamiento de intrapredicción 166 puede usar un modo de intrapredicción para generar los bloques predictivos de la PU en base a bloques de muestras espacialmente vecinas. La unidad de procesamiento de intrapredicción 166 puede determinar el modo de intrapredicción para la PU en base a uno o más elementos sintácticos obtenidos a partir del flujo de bits.

[0176] Si una PU se codifica usando interpredicción, la unidad de descodificación por entropía 150 puede determinar información de movimiento para la PU. La unidad de compensación de movimiento 164 puede determinar, en base a la información de movimiento de la PU, uno o más bloques de referencia. La unidad de compensación de movimiento 164 puede generar, en base a los uno o más bloques de referencia, bloques predictivos (por ejemplo, bloques predictivos de luma, Cb y Cr) para la PU.

[0177] La unidad de procesamiento de predicción 152 puede construir una primera lista de imágenes de referencia (RefPicList0) y una segunda lista de imágenes de referencia (RefPicList1) en base a elementos sintácticos extraídos del flujo de bits. Además, si una PU se codifica usando interpredicción, la unidad de descodificación por entropía 150 puede extraer información de movimiento para la PU. La unidad de compensación de movimiento 164 puede determinar, en base a la información de movimiento de la PU, una o más regiones de referencia para la PU. La unidad de compensación de movimiento 164 puede generar, en base a bloques de muestras en los uno o más bloques de referencia para la PU, bloques predictivos de luma, Cb y Cr para la PU.

[0178] La unidad de reconstrucción 158 puede usar bloques de transformada (por ejemplo, bloques de transformada de luma, Cb y Cr) para TU de una CU y los bloques predictivos (por ejemplo, bloques de luma, Cb y Cr) de las PU de la CU, es decir, cualquiera de los datos de intrapredicción o de los datos de interpredicción, según corresponda, para reconstruir los bloques de codificación (por ejemplo, bloques de codificación de luma, Cb y Cr) para la CU. Por ejemplo, la unidad de reconstrucción 158 puede agregar muestras de los bloques de transformada (por ejemplo, los bloques de transformada de luma, Cb y Cr) a las muestras correspondientes de los bloques predictivos (por ejemplo, los bloques predictivos de luma, Cb y Cr) para reconstruir los bloques de codificación (por ejemplo, bloques de codificación de luma, Cb y Cr) de la CU.

[0179] La unidad de filtro 160 puede realizar una operación de desbloqueo para reducir las distorsiones de bloqueo asociadas con los bloques de codificación de la CU. El descodificador de vídeo 30 puede almacenar los bloques de codificación de la CU en la memoria intermedia de imágenes descodificadas 162. La memoria intermedia de imágenes descodificadas 162 puede proporcionar imágenes de referencia para una posterior compensación de movimiento, intrapredicción y presentación en un dispositivo de visualización, tal como el dispositivo de visualización 32 de la FIG. 1. Por ejemplo, el descodificador de vídeo 30 puede realizar, en base a los bloques de la memoria intermedia de imágenes descodificadas 162, operaciones de intrapredicción o de interpredicción para PU de otras CU.

[0180] La FIG. 7 es un diagrama de flujo que ilustra una operación ejemplar para la codificación de datos de vídeo de acuerdo con una técnica de esta divulgación. La FIG. 7 al igual que los otros diagramas de flujo de esta divulgación se proporcionan como ejemplos. Otros ejemplos pueden incluir más, menos o diferentes acciones, o las acciones se pueden realizar en diferentes órdenes.

[0181] En el ejemplo de la FIG. 7, el codificador de vídeo 20 genera un bloque residual para una CU de una imagen de los datos de vídeo (200). Cada muestra en el bloque residual puede indicar una diferencia entre una muestra en un bloque predictivo de una PU de la CU y una muestra correspondiente en un bloque de codificación de la CU. Además, el codificador de vídeo 20 puede descomponer los bloques residuales de la CU en uno o más bloques de transformada (202). Una TU de la CU comprende un bloque de transformada de los uno o más bloques de transformada. En algunos ejemplos, el codificador de vídeo 20 puede usar la partición de árbol cuaternario para descomponer el bloque residual de la CU en los uno o más bloques de transformada. Además, en algunos ejemplos, el codificador de vídeo 20 aplica una o más transformadas a un bloque de transformada para que la TU genere un bloque de coeficientes para la TU que comprende niveles de coeficientes de la TU (204). Por ejemplo, el codificador de vídeo 20 puede aplicar una transformada de coseno discreta (DCT) al bloque de transformada. En otros ejemplos, el codificador de vídeo 20 omite la aplicación de la transformada. Por tanto, los niveles de coeficientes de la TU pueden ser valores residuales.

[0182] Adicionalmente, el codificador de vídeo 20 puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para codificar los niveles de coeficientes de

la TU (206). En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas.

[0183] La FIG. 8 es un diagrama de flujo que ilustra una operación ejemplar de descodificación de datos de vídeo de acuerdo con una técnica de esta divulgación. En el ejemplo de la FIG. 8, el descodificador de vídeo 30 usa un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficiente de una única TU de una CU actual de una imagen (250). En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas. Además, en este ejemplo, el descodificador de vídeo 30 puede reconstruir un bloque de codificación de la CU actual agregando muestras de una o más unidades de predicción de la CU actual a las muestras correspondientes de un bloque de transformada de la TU (252). Por ejemplo, el descodificador de vídeo 30 puede usar intrapredicción o interpredicción para determinar bloques predictivos de una o más PU de la CU actual. Adicionalmente, el descodificador de vídeo 30 puede aplicar una transformada inversa a los niveles de coeficientes de TU de la CU actual para determinar las muestras de los bloques de transformada de las TU de la CU actual. En algunos ejemplos, el descodificador de vídeo 30 puede cuantificar inversamente las muestras de los bloques de transformada antes de aplicar la transformada inversa.

[0184] La FIG. 9 es un diagrama de flujo que ilustra una operación ejemplar de descodificación de datos de vídeo de acuerdo con una técnica de esta divulgación. En el ejemplo de la FIG. 9, el descodificador de vídeo 30 puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficiente de una única TU de una CU actual de una imagen (300). En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas. Además, el descodificador de vídeo 30 puede cuantificar inversamente los niveles de coeficiente de la TU (302). Adicionalmente, el descodificador de vídeo 30 puede aplicar una transformada inversa a los niveles de coeficiente de la TU para reconstruir un bloque de transformada de la TU (304). El descodificador de vídeo 30 puede reconstruir un bloque de codificación de la CU actual agregando muestras de una o más unidades de predicción de la CU actual a muestras correspondientes de un bloque de transformada de la TU (306). Por ejemplo, el descodificador de vídeo 30 puede usar intrapredicción o interpredicción para determinar bloques predictivos de una o más PU de la CU actual. Adicionalmente, el descodificador de vídeo 30 puede aplicar una transformada inversa a los niveles de coeficientes de TU de la CU actual para determinar las muestras de los bloques de transformada de las TU de la CU actual. En algunos ejemplos, el descodificador de vídeo 30 puede cuantificar inversamente las muestras de los bloques de transformada antes de aplicar la transformada inversa.

[0185] La FIG. 10 es un diagrama de flujo que ilustra una operación ejemplar para la codificación de datos de vídeo de acuerdo con una técnica de esta divulgación. La operación de la FIG. 10 se puede usar como parte de las operaciones ejemplares de las FIGS. 7 a 9.

[0186] En el ejemplo de la FIG. 10, un codificador de vídeo obtiene (por ejemplo, si el codificador de vídeo es un descodificador de vídeo, tal como el descodificador de vídeo 30) o genera (por ejemplo, si el codificador de vídeo es un codificador de vídeo, tal como el codificador de vídeo 20) una serie de elementos sintácticos (350). Cada elemento sintáctico respectivo de la serie de elementos sintácticos indica un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU. En general, generar un elemento sintáctico puede comprender determinar un valor del elemento sintáctico y almacenar el valor del elemento sintáctico en un medio de almacenamiento legible por ordenador. Además, en el ejemplo de la FIG. 10, como parte del uso del primer procedimiento de derivación de parámetro de Rice y el segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficiente de la TU única (por ejemplo, en las acciones 204 de la FIG. 7, 250 de la FIG. 8 y 300 de la FIG. 9), el codificador de vídeo puede aplicar el primer procedimiento de derivación del parámetro de Rice a un elemento sintáctico que aparece primero en la TU en un orden de descodificación o análisis, donde el elemento sintáctico está en la serie de elementos sintácticos (352). En general, el orden de análisis es un orden en el que un proceso establece valores de sintaxis a partir de un flujo de bits binario codificado.

[0187] Adicionalmente, como parte del uso del primer procedimiento de derivación de parámetro de Rice y el segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficiente de la TU única (por ejemplo, en las acciones 204 de la FIG. 7, 250 de la FIG. 8 y 300 de la FIG. 9), el codificador de vídeo puede aplicar el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos (354). En algunos ejemplos, como parte de la aplicación del segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos, el codificador de vídeo puede usar una función para determinar un valor de parámetro de Rice para codificar un nivel de coeficiente de un coeficiente actual de la TU (356). Las entradas de la función pueden incluir un valor registrado y un valor de parámetro de Rice derivado actual. El valor registrado puede basarse en un valor de parámetro de Rice usado para descodificar un coeficiente previo al coeficiente actual. Además, el valor actual del parámetro derivado de Rice puede derivarse usando el procedimiento de derivación basado en plantillas. La función puede ser la función máxima.

[0188] La FIG. 11 es un diagrama de flujo que ilustra una operación ejemplar en la que se usa un procedimiento de derivación basado en estadísticas, de acuerdo con una técnica de esta divulgación. Un codificador de vídeo, tal como el codificador de vídeo 20 o el descodificador de vídeo 30, puede realizar la operación de la FIG. 11 como parte de la realización de las operaciones de cualquiera de las FIGS. 7 a 9. En el ejemplo de la FIG. 11, una imagen actual comprende una pluralidad de subbloques de 4x4 y el codificador de vídeo está configurado para obtener (por ejemplo, si el codificador de vídeo es un descodificador de vídeo, tal como el descodificador de vídeo 30) o generar (por ejemplo, si el codificador de vídeo es un codificador de vídeo, tal como el codificador de vídeo 20) un elemento sintáctico que indica un valor restante respectivo para un valor absoluto de un nivel de coeficiente de una TU de la imagen actual (400). Adicionalmente, como se analiza en otra parte de esta divulgación, el codificador de vídeo puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficientes de la TU. En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas.

[0189] Como parte del uso del primer procedimiento de derivación de parámetro de Rice (402), el codificador de vídeo puede dividir la pluralidad de subbloques de la imagen actual en una pluralidad de categorías de modo que, para subbloques respectivos de la pluralidad de subbloques, el subbloque respectivo se clasifica en base a si el subbloque respectivo es un bloque de omisión de transformada y si el subbloque respectivo es para un componente de luma (404).

[0190] Adicionalmente, para cada categoría respectiva de la pluralidad de categorías, el codificador de vídeo puede mantener un valor estadístico respectivo (por ejemplo, *statCoeff*) para la categoría respectiva (406). En algunos ejemplos, como parte del mantenimiento del valor estadístico respectivo para la categoría respectiva, el codificador de vídeo puede, para cada categoría respectiva de la pluralidad de categorías y para cada subbloque respectivo de la imagen que pertenece a la categoría respectiva, actualizar el valor estadístico respectivo para la categoría respectiva como máximo una vez para el subbloque respectivo usando un elemento sintáctico de nivel restante codificado primero para el subbloque respectivo. Como parte de la actualización del valor estadístico respectivo para la categoría respectiva, el codificador de vídeo puede incrementar el valor estadístico respectivo para la categoría respectiva si (absCoeffLevel >=3*(1<<(statCoeff/4))) y disminuir el valor estadístico respectivo para la categoría respectiva si ((2*absCoeffLevel)<(1<<(statCoeff/4))), donde absCoeffLevel es un nivel de coeficiente absoluto del subbloque respectivo y statCoeff es el valor estadístico respectivo para la categoría respectiva.

[0191] Para cada subbloque respectivo de la pluralidad de subbloques, el codificador de vídeo puede usar el valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo para inicializar un parámetro de Rice respectivo (408). El codificador de vídeo puede binarizar el elemento sintáctico usando un código Rice de K-ésimo orden, siendo K el parámetro de Rice determinado. En algunos ejemplos, como parte del uso del valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo para inicializar un parámetro de Rice respectivo para el subbloque respectivo, el codificador de vídeo puede, para cada subbloque respectivo de la pluralidad de subbloques, determinar el parámetro respectivo de Rice para el subbloque respectivo como mínimo de un parámetro máximo de Rice y el valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo dividido por 4.

[0192] La FIG. 12 es un diagrama de flujo que ilustra una operación ejemplar en la que se usa un procedimiento de derivación basado en plantillas, de acuerdo con una técnica de esta divulgación. Un codificador de vídeo, tal como el codificador de vídeo 20 o el descodificador de vídeo 30, puede realizar la operación de la FIG. 12 como parte de la realización de las operaciones de cualquiera de las FIGS. 7 a 9.

[0193] En el ejemplo de la FIG. 12, una imagen actual comprende una pluralidad de subbloques de 4x4, una plantilla local cubre vecinos de una muestra actual de una TU de una CU de la imagen actual, y la muestra actual está en una posición de escaneo actual. En el ejemplo de la FIG. 12, un codificador de vídeo, tal como el codificador de vídeo 20 o el descodificador de vídeo 30, puede obtener (por ejemplo, si el codificador de vídeo es un descodificador de vídeo, tal como el descodificador de vídeo 30) o generar (por ejemplo, si el codificador de vídeo es un codificador de vídeo, tal como el codificador de vídeo 20) un elemento sintáctico (por ejemplo, *coeff_abs_level_remaining*) que indica un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo para la muestra actual (450).

[0194] Como se analiza en otra parte de esta divulgación, el codificador de vídeo puede usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficientes de la TU. En este ejemplo, el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas y el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas. Como parte del uso del segundo procedimiento de derivación de parámetro de Rice (452), el codificador de vídeo puede, para cada vecino respectivo cubierto por la plantilla local, determinar un valor respectivo (por ejemplo, $\delta_i(x)$) para el vecino respectivo (454). En este ejemplo, el valor respectivo para el vecino respectivo es igual al valor absoluto del vecino menos 1 si el valor absoluto del vecino es mayor que 0 e igual a 0 si el vecino es igual a 0. Adicionalmente, el codificador de vídeo puede determinar un valor de suma (por ejemplo, *sum_absolute_levelMinus1*) igual a una suma de valores para los vecinos (456). Además, en el ejemplo de

la FIG. 12, el codificador de vídeo puede determinar que un parámetro de Rice es igual a 0 si el valor de la suma cae en un primer intervalo, igual a 1 si el valor de la suma cae en un segundo intervalo, igual a 2 si el valor de la suma cae en un tercer intervalo, e igual a 3 si el valor de la suma cae en un cuarto intervalo (458). El elemento sintáctico se binariza usando un código Rice de K-ésimo orden, siendo K igual al parámetro de Rice determinado.

[0195] La FIG. 13 es un diagrama de flujo que ilustra una operación ejemplar para determinar un parámetro de Rice basado en una función genérica, de acuerdo con una técnica de esta divulgación. La operación de la FIG. 13 se puede usar por separado o junto con otros ejemplos de esta divulgación. La operación de la FIG. 13 puede ser una forma de realizar el proceso de derivación de parámetro de Rice basado en plantillas descrito en otros diagramas de flujo de esta divulgación.

[0196] En el ejemplo de la FIG. 13, una imagen actual comprende una pluralidad de subbloques de 4x4, una plantilla local cubre vecinos de una muestra actual de una TU de una CU de la imagen actual, y la muestra actual está en una posición de escaneo actual. Como parte del uso de un procedimiento de derivación de parámetro de Rice basado en plantillas, un codificador de vídeo puede, para cada vecino respectivo cubierto por la plantilla local, determinar un valor respectivo para el vecino respectivo (500). El valor respectivo para el vecino respectivo es igual al valor absoluto del vecino menos 1 si el valor absoluto del vecino es mayor que 0 e igual a 0 si el vecino es igual a 0. Además, el codificador de vídeo puede determinar un valor de suma igual a una suma de valores para los vecinos (502). Adicionalmente, el codificador de vídeo puede determinar un valor K (504). Un nivel de coeficiente de la TU se binariza usando un código Rice de K-ésimo orden. Por ejemplo, el codificador de vídeo 20 puede binarizar el nivel de coeficiente y el descodificador de vídeo 30 puede desbinarizar el nivel de coeficiente.

[0197] En este ejemplo, se puede seleccionar una definición de K de un grupo que consiste en:

- K que se define como un número entero mínimo que satisface (1 << (K + 3)) > (*sum absolute_levelMinus1* + M), donde M es un número entero y sum_absolute_levelminus1 es el valor de la suma,

- K que se define como un número entero mínimo que satisface (1 << (K + 3)) > (*sum absolute_levelMinus1* + M), donde M es un número entero y sum_absolute_levelMinus1 es el valor de la suma, y

- K que se define como un número entero mínimo que satisface (1 << (K + 3)) ≥ (*sum_absolute_levelMinus1* + M), donde M es un número entero y sum_absolute_levelMinus1 es el valor de la suma.

[0198] La FIG. 14 es un diagrama de flujo que ilustra una operación ejemplar para binarizar o desbinarizar una serie de elementos sintácticos, de acuerdo con una técnica de esta divulgación. La operación de la FIG. 14 se puede usar por separado o junto con otros ejemplos de esta divulgación. En el ejemplo de la FIG. 14, un codificador de vídeo (por ejemplo, el codificador de vídeo 20 o el descodificador de vídeo 30) genera u obtiene una serie de elementos sintácticos (por ejemplo, elementos sintácticos *abs_coeff_level_remaining*) (550). En el ejemplo de la FIG. 14, cada elemento sintáctico respectivo de la serie de elementos sintácticos puede indicar un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU.

[0199] Además, en el ejemplo de la FIG. 14, el codificador de vídeo puede realizar las acciones 552-560 para cada elemento sintáctico respectivo de la serie de elementos sintácticos (551). En el ejemplo de la FIG. 14, el codificador de vídeo puede derivar un parámetro K de Rice para el elemento sintáctico respectivo usando un primer procedimiento de derivación de parámetro de Rice (por ejemplo, un procedimiento de derivación de parámetro de Rice basado en estadísticas) o un segundo procedimiento de derivación de parámetro de Rice (por ejemplo, un procedimiento de derivación de parámetro de Rice basado en plantillas) (552). Además, el codificador de vídeo puede establecer un punto de conmutación para el elemento sintáctico respectivo igual a (M << K) (554). M depende de K. Por ejemplo, el codificador de vídeo puede establecer M igual a 6, 5 y 6 para valores de K iguales a 0, 1 y 2, respectivamente, y puede establecer M a 3 para todos los demás valores de K.

[0200] Además, en el ejemplo de la FIG. 14, el codificador de vídeo puede determinar si el elemento sintáctico respectivo es menor que el punto de conmutación (556). En respuesta a la determinación de que el elemento sintáctico respectivo es menor que el punto de conmutación (rama "SÍ" de 556), el codificador de vídeo puede binarizar o desbinarizar el elemento sintáctico respectivo usando un código Rice de K-ésimo orden (558). De lo contrario, si el elemento sintáctico respectivo es menor que el punto de conmutación (rama "NO" de 556), el codificador de vídeo puede binarizar o desbinarizar el elemento sintáctico respectivo usando un código Exp-Golom de K-ésimo orden (560).

[0201] Debería entenderse que todas las técnicas descritas en el presente documento pueden usarse individualmente o en combinación. Se debe reconocer que, dependiendo del ejemplo, determinadas acciones o eventos de cualquiera de las técnicas descritas en el presente documento se pueden realizar en una secuencia distinta, se pueden añadir, fusionar o excluir por completo (por ejemplo, no todas las acciones o eventos descritos son necesarios para la puesta en práctica de las técnicas). Por otro lado, en determinados ejemplos, las acciones o eventos pueden tener lugar simultáneamente, por ejemplo, a través de procesamiento de múltiples subprocesos, procesamiento de interrupciones o múltiples procesadores, en lugar de secuencialmente. Además, aunque con propósitos de claridad se describe que un único módulo o unidad realiza determinados aspectos de esta divulgación,

se debería entender que una combinación de unidades o módulos asociados a un codificador de vídeo pueden realizar las técnicas de esta divulgación.

[0202] Determinados aspectos de esta divulgación se han descrito con respecto a la norma de HEVC con fines ilustrativos. Sin embargo, las técnicas descritas en esta divulgación pueden ser útiles para otros procesos de codificación de vídeo, que incluyen otros procesos de codificación de vídeo, estándar o de propiedad, aún no desarrollados.

[0203] El codificador de vídeo 20 (FIGS. 1 y 5) y/o el descodificador de vídeo 30 (FIGS. 1 y 6), los cuales se pueden denominar en general codificadores de vídeo, pueden realizar las técnicas descritas anteriormente. Asimismo, la codificación de vídeo se puede referir a una codificación de vídeo o una descodificación de vídeo, según corresponda. Además, aunque esta divulgación se ha referido al elemento sintáctico *coeff_abs_level remaining*, las técnicas de esta divulgación pueden aplicarse a elementos sintácticos con nombres diferentes con la misma semántica que el elemento sintáctico *coeff_abs_level remaining* u otros elementos sintácticos de una unidad de transformada.

[0204] Aunque se han descrito anteriormente combinaciones particulares de diversos aspectos de las técnicas, estas combinaciones se proporcionan simplemente para ilustrar ejemplos de las técnicas descritas en esta divulgación. En consecuencia, las técnicas de esta divulgación no se deben limitar a estos ejemplos de combinaciones, sino que pueden abarcar cualquier combinación concebible de los diversos aspectos de las técnicas descritas en esta divulgación.

[0205] En uno o más ejemplos, las funciones descritas se pueden implementar en hardware, software, firmware o en cualquier combinación de los mismos. Si se implementan en software, las funciones se pueden almacenar en o transmitir sobre un medio legible por ordenador como una o más instrucciones o código, y ejecutarse por una unidad de procesamiento basada en hardware. Los medios legibles por ordenador pueden incluir medios de almacenamiento legibles por ordenador, que corresponden a un medio tangible tal como medios de almacenamiento de datos, o medios de comunicación que incluyen cualquier medio que facilita la transferencia de un programa informático de un lugar a otro, por ejemplo, de acuerdo con un protocolo de comunicación. De esta manera, los medios legibles por ordenador pueden corresponder, en general, a (1) medios de almacenamiento tangibles legibles por ordenador que son no transitorios o (2) un medio de comunicación tal como una señal o una onda portadora. Los medios de almacenamiento de datos pueden ser cualquier medio disponible al que se pueda acceder desde uno o más ordenadores o uno o más procesadores para recuperar instrucciones, código y/o estructuras de datos para la implementación de las técnicas descritas en la presente divulgación. Un producto de programa informático puede incluir un medio legible por ordenador.

[0206] A modo de ejemplo, y no de limitación, dichos medios de almacenamiento legibles por ordenador pueden comprender RAM, ROM, EEPROM, CD-ROM u otro almacenamiento en disco óptico, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, memoria flash o cualquier otro medio que se pueda usar para almacenar código de programa deseado en forma de instrucciones o estructuras de datos y al que se pueda acceder mediante un ordenador. Además, cualquier conexión recibe apropiadamente la denominación de medio legible por ordenador. Por ejemplo, si las instrucciones se transmiten desde un sitio web, un servidor u otra fuente remota usando un cable coaxial, un cable de fibra óptica, un par trenzado, una línea de abonado digital (DSL) o tecnologías inalámbricas tales como infrarrojos, radio y microondas, entonces el cable coaxial, el cable de fibra óptica, el par trenzado, la DSL o las tecnologías inalámbricas tales como infrarrojos, radio y microondas están incluidos en la definición de medio. Sin embargo, se debe entender que los medios de almacenamiento legibles por ordenador y los medios de almacenamiento de datos no incluyen conexiones, ondas portadoras, señales ni otros medios transitorios, sino que, en cambio, se refieren a medios de almacenamiento tangibles no transitorios. El término disco, como se usa en el presente documento, incluye disco compacto (CD), disco láser, disco óptico, disco versátil digital (DVD), disco flexible y disco Blu-ray, donde unos discos reproducen normalmente los datos magnéticamente, mientras que otros discos reproducen datos ópticamente con láseres. Las combinaciones de lo anterior también se deben incluir dentro del alcance de los medios legibles por ordenador.

[0207] Uno o más procesadores, tales como uno o más procesadores de señales digitales (DSP), microprocesadores de uso general, circuitos integrados específicos de la aplicación (ASIC), matrices lógicas programables por campo (FPGA) u otros circuitos lógicos integrados o discretos equivalentes pueden ejecutar las instrucciones. En consecuencia, el término "procesador", como se usa en el presente documento, se puede referir a cualquiera de las estructuras anteriores o a cualquier otra estructura adecuada para la implementación de las técnicas descritas en el presente documento. Además, en algunos aspectos, la funcionalidad descrita en el presente documento se puede proporcionar dentro de módulos de hardware y/o de software dedicados configurados para codificar y descodificar, o incorporar en un códec combinado. Además, las técnicas se podrían implementar por completo en uno o más circuitos o elementos lógicos.

[0208] Las técnicas de la presente divulgación se pueden implementar en una amplia variedad de dispositivos o aparatos, incluyendo un teléfono inalámbrico, un circuito integrado (IC) o un conjunto de IC (por ejemplo, un conjunto de chips). En la presente divulgación se describen diversos componentes, módulos o unidades para destacar aspectos funcionales de dispositivos configurados para realizar las técnicas divulgadas, pero no se requiere necesariamente su

realización mediante diferentes unidades de hardware. En su lugar, como se describe anteriormente, diversas unidades se pueden combinar en una unidad de hardware de códec o proporcionar mediante un grupo de unidades de hardware interoperativas, que incluya uno o más procesadores como se describe anteriormente, junto con software y/o firmware adecuados.

[0209] Se han descrito diversos ejemplos. Estos y otros ejemplos están dentro del alcance de las siguientes reivindicaciones.

REIVINDICACIONES

1. Un procedimiento de descodificación de datos de vídeo, comprendiendo el procedimiento:

usar (300) un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficientes de una sola unidad de transformada (TU) de una unidad de codificación actual (CU) de una imagen actual de los datos de vídeo, en el que:

el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas, y

el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas, en el que el procedimiento de derivación basado en plantillas determina un parámetro de Rice respectivo para un elemento sintáctico respectivo de los niveles de coeficiente basados en elementos sintácticos vecinos de niveles de coeficientes vecinos cubiertos por una plantilla local;

reconstruir (306) un bloque de codificación de la CU actual agregando muestras de una o más unidades de predicción de la CU actual a las muestras correspondientes de un bloque de transformada de la TU;

obtener (350) una serie de elementos sintácticos, indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU, en el que el uso del primer procedimiento de derivación de parámetro de Rice y el segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficientes de la TU única comprende:

aplicar (352) el primer procedimiento de derivación de parámetro de Rice a un elemento sintáctico que aparece primero en la TU en un orden de descodificación o análisis, en el que el elemento sintáctico está en la serie de elementos sintácticos; y

aplicar (354) el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos, **caracterizado por que**

aplicar (356) el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos comprende el uso de una función para determinar un valor de parámetro de Rice para descodificar un nivel de coeficiente de un coeficiente actual de la TU, en el que las entradas de la función incluyen un valor registrado y un valor actual derivado del parámetro de Rice, basándose el valor registrado en un valor del parámetro de Rice usado para descodificar un coeficiente anterior al coeficiente actual, usando el valor actual derivado del parámetro de Rice el procedimiento de derivación basado en plantillas; y

establecer el valor registrado igual al valor del parámetro de Rice determinado para descodificar el nivel de coeficiente del coeficiente actual, menos 1.

2. El procedimiento de la reivindicación 1, que comprende además:

basado en que el valor registrado sea menor que 0, restablecer el valor registrado a 0.

3. El procedimiento de la reivindicación 1, en el que el uso del segundo procedimiento de derivación de parámetro de Rice comprende:

determinar un valor igual a una suma absoluta de cada nivel de una pluralidad de niveles de coeficiente menos 1, en el que cada nivel de coeficiente respectivo de la pluralidad de niveles de coeficiente está en una región definida por una plantilla;

determinar un parámetro de Rice basado en el valor; y

generar, basado en el parámetro de Rice, un valor descodificado para un nivel de coeficiente actual de la TU.

4. El procedimiento de la reivindicación 1, que comprende además:

obtener una serie de elementos sintácticos, indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU,

para cada respectivo elemento sintáctico de la serie de elementos sintácticos:

derivar un parámetro K de Rice para el elemento sintáctico respectivo usando el primer procedimiento de derivación de parámetro de Rice o el segundo procedimiento de derivación de parámetro de Rice, y

establecer un punto de conmutación para el elemento sintáctico respectivo igual a (M << K), en el que:

M depende de K,

si el elemento sintáctico respectivo es menor que el punto de conmutación, el elemento sintáctico respectivo se binariza usando un código Rice de K-ésimo orden, y

si el elemento sintáctico respectivo es mayor o igual que el punto de conmutación, el elemento sintáctico respectivo se binariza usando un prefijo y un sufijo usando un código Exp-Golomb de K-ésimo orden.

5. El procedimiento de la reivindicación 1, que comprende además:

cuantificar de forma inversa los niveles de coeficientes de la TU; y

aplicar una transformada inversa a los niveles de coeficiente de la TU para reconstruir el bloque de transformada de la TU.

6. Un procedimiento de codificación de datos de vídeo, comprendiendo el procedimiento:

generar (200) un bloque residual para una unidad de codificación (CU) de una imagen actual de los datos de vídeo, indicando cada muestra en el bloque residual una diferencia entre una muestra en un bloque predictivo para una unidad de predicción (PU) de la CU y una muestra correspondiente en un bloque de codificación de la CU;

descomponer (202) el bloque residual para la CU en uno o más bloques de transformada, en el que una unidad de transformada (TU) de la CU comprende un bloque de transformada de uno o más bloques de transformada; y

usar (206) un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para codificar niveles de coeficientes de la TU, en el que:

el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas, y

el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas, en el que el procedimiento de derivación basado en plantillas determina un parámetro de Rice respectivo para un elemento sintáctico respectivo de los niveles de coeficiente basados en elementos sintácticos vecinos de niveles de coeficientes vecinos cubiertos por una plantilla local.

generar (350) una serie de elementos sintácticos, indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU; y

en el que el uso del primer procedimiento de derivación de parámetro de Rice y el segundo procedimiento de derivación de parámetro de Rice para codificar niveles de coeficientes de la TU única comprende:

aplicar (352) el primer procedimiento de derivación de parámetro de Rice a un elemento sintáctico que aparece primero en la TU en un orden de descodificación o análisis, en el que el elemento sintáctico está en la serie de elementos sintácticos; y

aplicar (354) el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos; **caracterizado por que**

aplicar el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos comprende:

usar (356) una función para determinar un valor de parámetro de Rice para codificar un nivel de coeficiente de un coeficiente actual de la TU, en el que las entradas de la función incluyen un valor registrado y un valor actual derivado del parámetro de Rice, basándose el valor registrado en un valor del parámetro de Rice usado para descodificar un coeficiente anterior al coeficiente actual, usando el valor actual derivado del parámetro de Rice

el procedimiento de derivación basado en plantillas; y

establecer el valor registrado igual al valor del parámetro de Rice determinado para descodificar el nivel de coeficiente del coeficiente actual, menos 1.

7. El procedimiento de la reivindicación 6, en el que el uso del segundo procedimiento de derivación de parámetro de Rice comprende:

determinar un valor igual a una suma absoluta de cada nivel de una pluralidad de niveles de coeficiente menos 1, en el que cada nivel de coeficiente respectivo de la pluralidad de niveles de coeficiente está en una región definida por una plantilla;

determinar un parámetro de Rice basado en el valor; y

generar, basado en el parámetro de Rice, un valor codificado para un nivel de coeficiente actual de la TU; y que preferentemente comprende además:

generar una serie de elementos sintácticos, indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU; y

para cada respectivo elemento sintáctico de la serie de elementos sintácticos:

derivar un parámetro K de Rice para el elemento sintáctico respectivo usando el primer procedimiento de derivación de parámetro de Rice o el segundo procedimiento de derivación de parámetro de Rice,

establecer un punto de conmutación para el elemento sintáctico respectivo igual a (M << K), en el que:

M depende de K,

para cada respectivo elemento sintáctico de la serie de elementos sintácticos:

si el elemento sintáctico respectivo es menor que el punto de conmutación, binarizar el elemento sintáctico respectivo usando un código Rice de K-ésimo orden, y

si el elemento sintáctico respectivo es mayor o igual que el punto de conmutación, binarizar el elemento sintáctico respectivo usando un prefijo y un sufijo usando un código Exp-Golomb de K-ésimo orden.

8. El procedimiento de la reivindicación 6, en el que:

descomponer el bloque residual para la CU en uno o más bloques de transformada comprende usar la partición de árbol cuaternario para descomponer el bloque residual para la CU en uno o más bloques de transformada, y

el procedimiento comprende además aplicar una o más transformadas al bloque de transformada para que la TU genere un bloque de coeficientes para la TU, comprendiendo el bloque de coeficientes para la TU los niveles de coeficientes de la TU.

9. Un dispositivo para codificar datos de vídeo, comprendiendo el dispositivo:

un medio de almacenamiento legible por ordenador configurado para almacenar los datos de vídeo; y

uno o más procesadores configurados para:

usar un primer procedimiento de derivación de parámetro de Rice y un segundo procedimiento de derivación de parámetro de Rice para codificar niveles de coeficientes de una sola unidad de transformada (TU) de una unidad de codificación actual (CU) de una imagen actual de los datos de vídeo, en el que:

el primer procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en estadísticas, y

el segundo procedimiento de derivación de parámetro de Rice es un procedimiento de derivación basado en plantillas, en el que el procedimiento de derivación basado en plantillas determina un parámetro de Rice respectivo para un elemento sintáctico respectivo de los niveles de coeficiente basados en elementos sintácticos vecinos de niveles de coeficientes vecinos cubiertos por una plantilla local;

obtener o generar una serie de elementos sintácticos, indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU; y

en el que uno o más procesadores están configurados de modo que, como parte del uso del primer procedimiento de derivación de parámetro de Rice y el segundo procedimiento de derivación de parámetro de Rice para descodificar niveles de coeficiente de la TU única, los uno o más procesadores:

aplican el primer procedimiento de derivación de parámetro de Rice a un elemento sintáctico que aparece primero en la TU en un orden de descodificación o análisis, en el que el elemento sintáctico está en la serie de elementos sintácticos; y

aplican el segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos;

caracterizado por que

como parte de la aplicación del segundo procedimiento de derivación de parámetro de Rice a cada uno de los otros elementos sintácticos de la serie de elementos sintácticos, los uno o más procesadores:

usan una función para determinar un valor de parámetro de Rice para codificar un nivel de coeficiente de un coeficiente actual de la TU, en el que las entradas de la función incluyen un valor registrado y un valor actual derivado del parámetro de Rice, basándose el valor registrado en un valor del parámetro de Rice usado para descodificar un coeficiente anterior al coeficiente actual, usando el valor actual derivado del parámetro de Rice el procedimiento de derivación basado en plantillas; y

establecen el valor registrado igual al valor del parámetro de Rice determinado para descodificar el nivel de coeficiente del coeficiente actual, menos 1.

10. El dispositivo de la reivindicación 9, en el que los uno o más procesadores están configurados para:

basado en que el valor registrado sea menor que 0, restablecer el valor registrado a 0.

11. El dispositivo de la reivindicación 9, en el que los uno o más procesadores están configurados de modo que, como parte del uso del primer procedimiento de derivación de parámetro de Rice, los uno o más procesadores:

determinan un valor igual a una suma absoluta de cada nivel de una pluralidad de niveles de coeficiente menos 1, en el que cada nivel de coeficiente respectivo de la pluralidad de niveles de coeficiente está en una región definida por una plantilla;

determinan un parámetro de Rice basado en el valor; y

generan, basado en el parámetro de Rice, un valor codificado para un nivel de coeficiente actual de la TU.

12. El dispositivo de la reivindicación 11, en el que la imagen actual comprende una pluralidad de subbloques de 4x4, la plantilla cubre los vecinos de una muestra actual de la TU, estando la muestra actual en una posición de escaneo actual, y los uno o más procesadores están configurados de modo que, como parte del uso del segundo procedimiento de derivación de parámetro de Rice, los uno o más procesadores:

para cada vecino respectivo cubierto por la plantilla local, determinan un valor respectivo para el vecino respectivo, siendo el valor respectivo para el vecino respectivo igual a un valor absoluto del vecino menos 1 si el valor absoluto del vecino es mayor que 0 e igual a 0 si el vecino es igual a 0;

determinan un valor de suma igual a una suma de valores para los vecinos; y

determinan un valor K, en el que se binariza un nivel de coeficiente de la TU usando un código Rice de K-ésimo orden y se selecciona una definición de K de un grupo que consiste en:

K que se define como un número entero mínimo que satisface

(1 << (K + 3)) > (*sum_absolute_levelMinus1* + M), donde M es un número entero y sum_absolute_levelMinus1 es el valor de la suma, y

K que se define como un número entero mínimo que satisface

(1 << (K + 3)) ≥ (*sum_absolute_levelMinus1* + M), donde M es un número entero y sum_absolute_levelMinus1 es el valor de la suma; y preferentemente en el que K está limitado a un umbral.

13. El dispositivo de la reivindicación 9, en el que los uno o más procesadores están configurados para:

obtener o generar una serie de elementos sintácticos, indicando cada elemento sintáctico respectivo de la serie de elementos sintácticos un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo de la TU, y

para cada respectivo elemento sintáctico de la serie de elementos sintáctico, los uno o más procesadores están configurados para:

derivar un parámetro K de Rice para el elemento sintáctico respectivo usando el primer procedimiento de derivación de parámetro de Rice o el segundo procedimiento de derivación de parámetro de Rice, y

establecer un punto de conmutación para el elemento sintáctico respectivo igual a (M << K), en el que:

M depende de K,

para cada respectivo elemento sintáctico de la serie de elementos sintácticos:

si el elemento sintáctico respectivo es menor que el punto de conmutación, el elemento sintáctico respectivo se binariza usando un código Rice de K-ésimo orden, y

si el elemento sintáctico respectivo es mayor o igual que el punto de conmutación, el elemento sintáctico respectivo se binariza usando un prefijo y un sufijo usando un código Exp-Golomb de K-ésimo orden; o en el que

la imagen actual comprende una pluralidad de subbloques de 4x4,

los uno o más procesadores están configurados para obtener o generar un elemento sintáctico que indica un valor restante respectivo para un valor absoluto de un nivel de coeficiente de la TU, y

los uno o más procesadores se configuran de modo que, como parte del uso del primer procedimiento de derivación de parámetro de Rice, los uno o más procesadores:

dividen la pluralidad de subbloques de una imagen actual en una pluralidad de categorías de modo que, para subbloques respectivos de la pluralidad de subbloques, el subbloque respectivo se clasifica en base a si el subbloque respectivo es un bloque de omisión de transformada y si el subbloque respectivo es para un componente de luma;

para cada categoría respectiva de la pluralidad de categorías, mantienen un valor estadístico respectivo para la categoría respectiva; y

para cada subbloque respectivo de la pluralidad de subbloques, usan el valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo para inicializar un parámetro de Rice respectivo para el subbloque respectivo,

en el que el nivel de coeficiente está en un subbloque particular de la pluralidad de subbloques de 4x4 y el elemento sintáctico se binariza usando un código Rice de K-ésimo orden, siendo K el parámetro respectivo de Rice para el subbloque particular.

14. El dispositivo de la reivindicación 13, en el que:

para cada categoría respectiva de la pluralidad de categorías, los uno o más procesadores están configurados de modo que, como parte del mantenimiento del valor estadístico respectivo para la categoría respectiva, los uno o más procesadores:

para cada subbloque respectivo de la imagen actual que pertenece a la categoría respectiva, actualizan el valor estadístico respectivo para la categoría respectiva como máximo una vez para el subbloque respectivo usando

un elemento sintáctico de nivel restante codificado primero para el subbloque respectivo, en el que los uno o más procesadores están configurados de modo que, como parte de la actualización del valor estadístico respectivo para la categoría respectiva, los uno o más procesadores: incrementan el valor estadístico respectivo para la categoría respectiva si

$$(absCoeffLevel >= 3*(1<<(statCoeff /4)));$$

y disminuyen el valor estadístico respectivo para la categoría respectiva si

$$((2*absCoeffLevel)<(1<<(statCoeff/4))),$$

en el que absCoeffLevel es un nivel de coeficiente absoluto del subbloque respectivo y statCoeff es el valor estadístico respectivo para la categoría respectiva; y

para cada subbloque respectivo de la pluralidad de subbloques, los uno o más procesadores se configuran de modo que, como parte del uso del valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo, para inicializar un parámetro de Rice respectivo para el subbloque respectivo, los uno o más procesadores:

determinan el parámetro respectivo de Rice para el subbloque respectivo como mínimo de un parámetro máximo de Rice y el valor estadístico respectivo para la categoría a la que pertenece el subbloque respectivo dividido por 4.

15. El dispositivo de la reivindicación 9, en el que:

la imagen actual comprende una pluralidad de subbloques de 4x4,

una plantilla local cubre los vecinos de una muestra actual de la TU,

la muestra actual está en una posición de escaneo actual,

los uno o más procesadores están configurados para obtener o generar un elemento sintáctico que indica un valor restante respectivo para un valor absoluto de un nivel de coeficiente respectivo para la muestra actual, y

los uno o más procesadores se configuran de modo que, como parte del uso del segundo procedimiento de derivación de parámetro de Rice, los uno o más procesadores:

para cada vecino respectivo cubierto por la plantilla local, determinan un valor respectivo para el vecino respectivo, siendo el valor respectivo para el vecino respectivo igual a un valor absoluto del vecino menos 1 si el valor absoluto del vecino es mayor que 0 e igual a 0 si el vecino es igual a 0;

determinan un valor de suma igual a una suma de valores para los vecinos;

determinan que un parámetro de Rice es igual a 0 si el valor de la suma cae en un primer intervalo, igual a 1 si el valor de la suma cae en un segundo intervalo, igual a 2 si el valor de la suma cae en un tercer intervalo, e igual a 3 si el valor de la suma cae en un cuarto intervalo; y

en el que el elemento sintáctico se binariza usando un código Rice de K-ésimo orden, siendo K igual al parámetro de Rice determinado.

16. El dispositivo de la reivindicación 9, en el que los uno o más procesadores están configurados para:

generar un bloque residual para la CU, indicando cada muestra en el bloque residual una diferencia entre una muestra en un bloque predictivo para una unidad de predicción (PU) de la CU y una muestra correspondiente en un bloque de codificación de la CU;

descomponer el bloque residual para la CU en uno o más bloques de transformada, en el que la TU comprende un bloque de transformada de uno o más bloques de transformada; y

aplicar una o más transformadas al bloque de transformada para generar un bloque de coeficientes para la TU, comprendiendo el bloque de coeficientes para la TU los niveles de coeficientes de la TU; o

en el que los uno o más procesadores están configurados para:

cuantificar de forma inversa los niveles de coeficientes de la TU; y

aplicar una transformada inversa a los niveles de coeficiente de la TU para reconstruir el bloque de transformada de la TU; y

reconstruir un bloque de codificación de la CU actual agregando muestras de una o más unidades de predicción de la CU actual a las muestras correspondientes de un bloque de transformada de la TU.

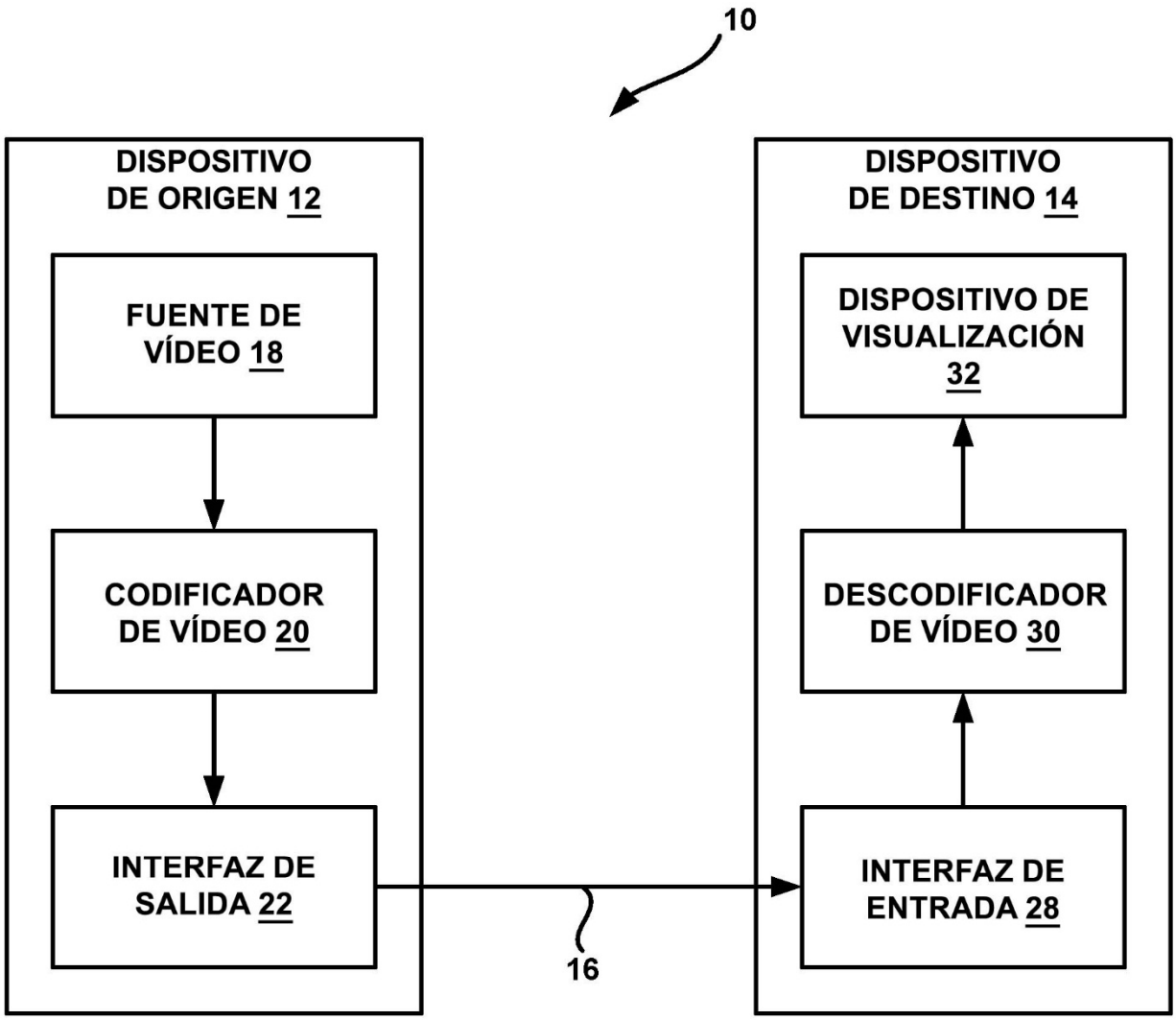
10
DISPOSITIVO DE ORIGEN 12
FUENTE DE VÍDEO 18
CODIFICADOR DE VÍDEO 20
INTERFAZ DE SALIDA 22
16
DISPOSITIVO DE DESTINO 14
DISPOSITIVO DE VISUALIZACIÓN 32
DESCODIFICADOR DE VÍDEO 30
INTERFAZ DE ENTRADA 28

FIG. 1

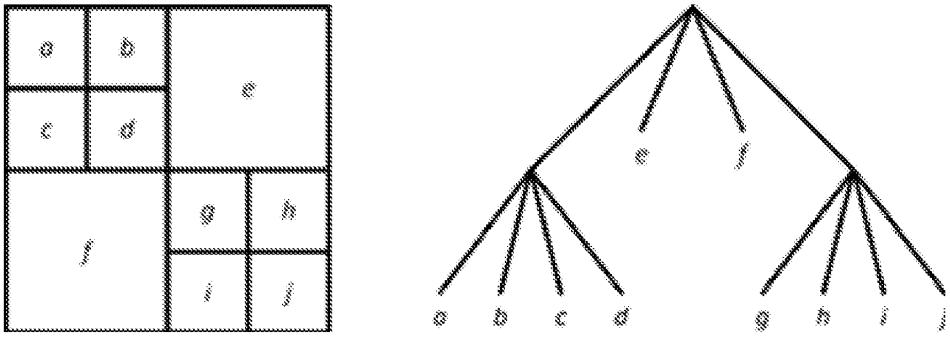

FIG. 2

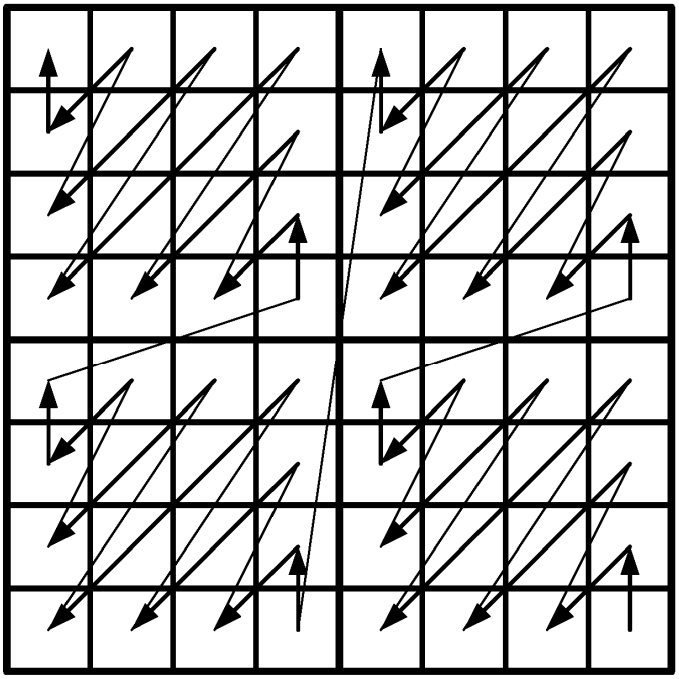

FIG. 3

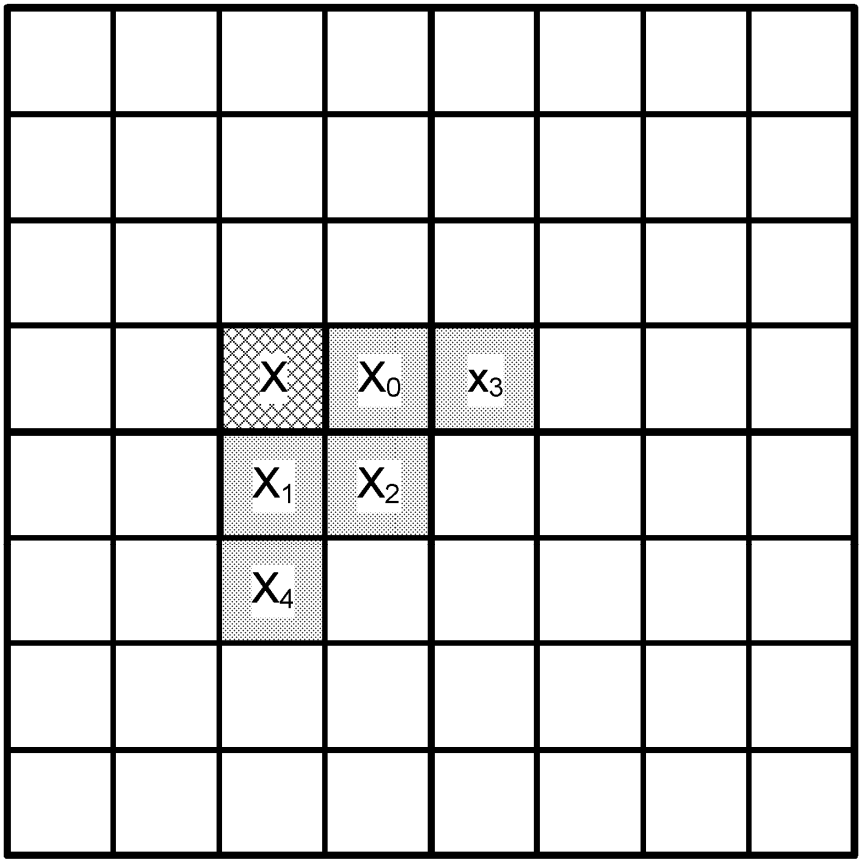

FIG. 4

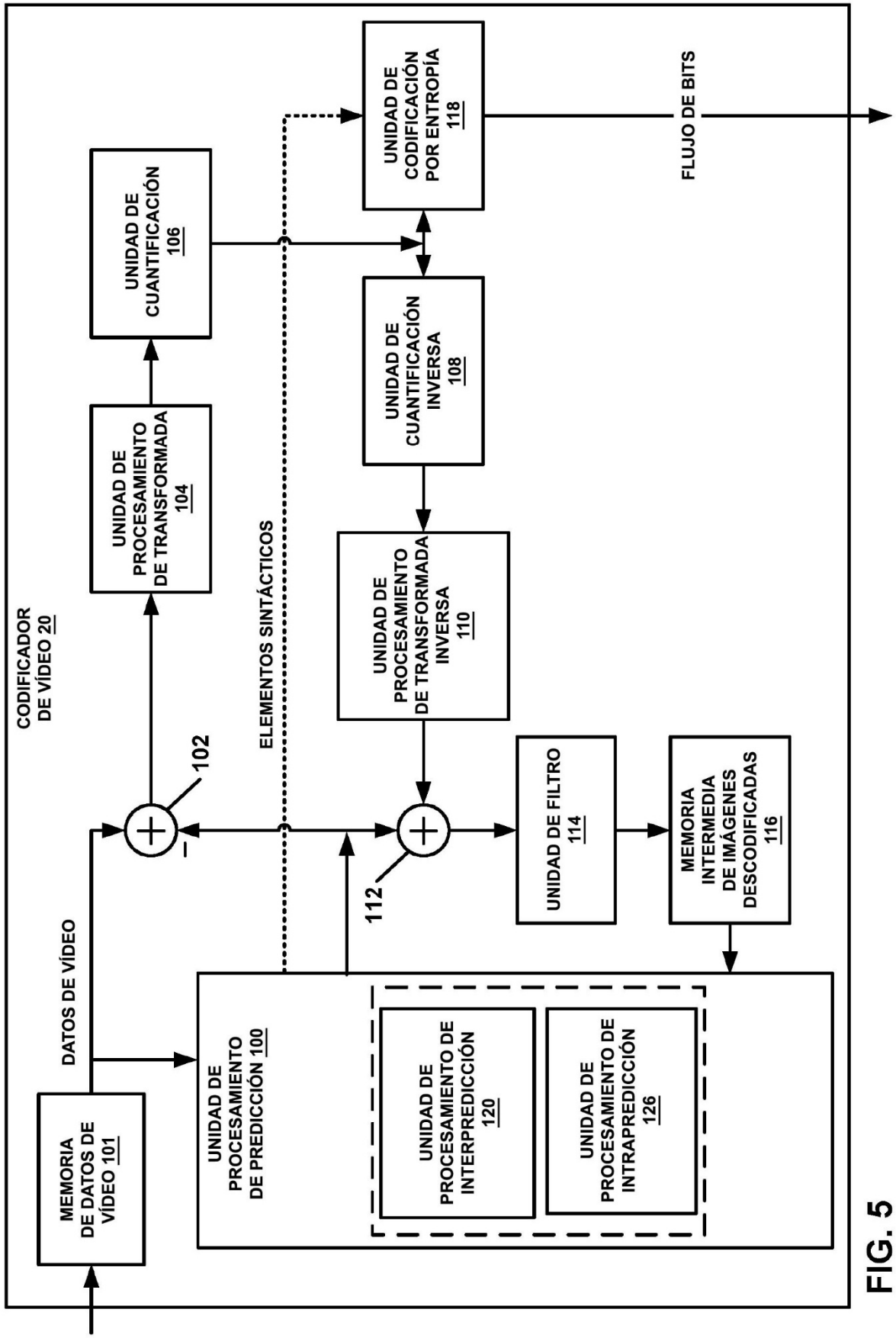
CODIFICADOR DE VÍDEO 20
MEMORIA DE DATOS DE VÍDEO 101
DATOS DE VÍDEO
UNIDAD DE PROCESAMIENTO DE PREDICCIÓN 100
UNIDAD DE PROCESAMIENTO DE INTERPREDICCIÓN 120
UNIDAD DE PROCESAMIENTO DE INTRAPREDICCIÓN 126
UNIDAD DE PROCESAMIENTO DE TRANSFORMADA 104
UNIDAD DE CUANTIFICACIÓN 106
UNIDAD DE CUANTIFICACIÓN INVERSA 108
UNIDAD DE PROCESAMIENTO DE TRANSFORMADA INVERSA 110
UNIDAD DE CODIFICACIÓN POR ENTROPÍA 118
ELEMENTOS SINTÁCTICOS
FLUJO DE BITS
102
112
UNIDAD DE FILTRO 114
MEMORIA INTERMEDIA DE IMÁGENES DESCODIFICADAS 116

FIG. 5

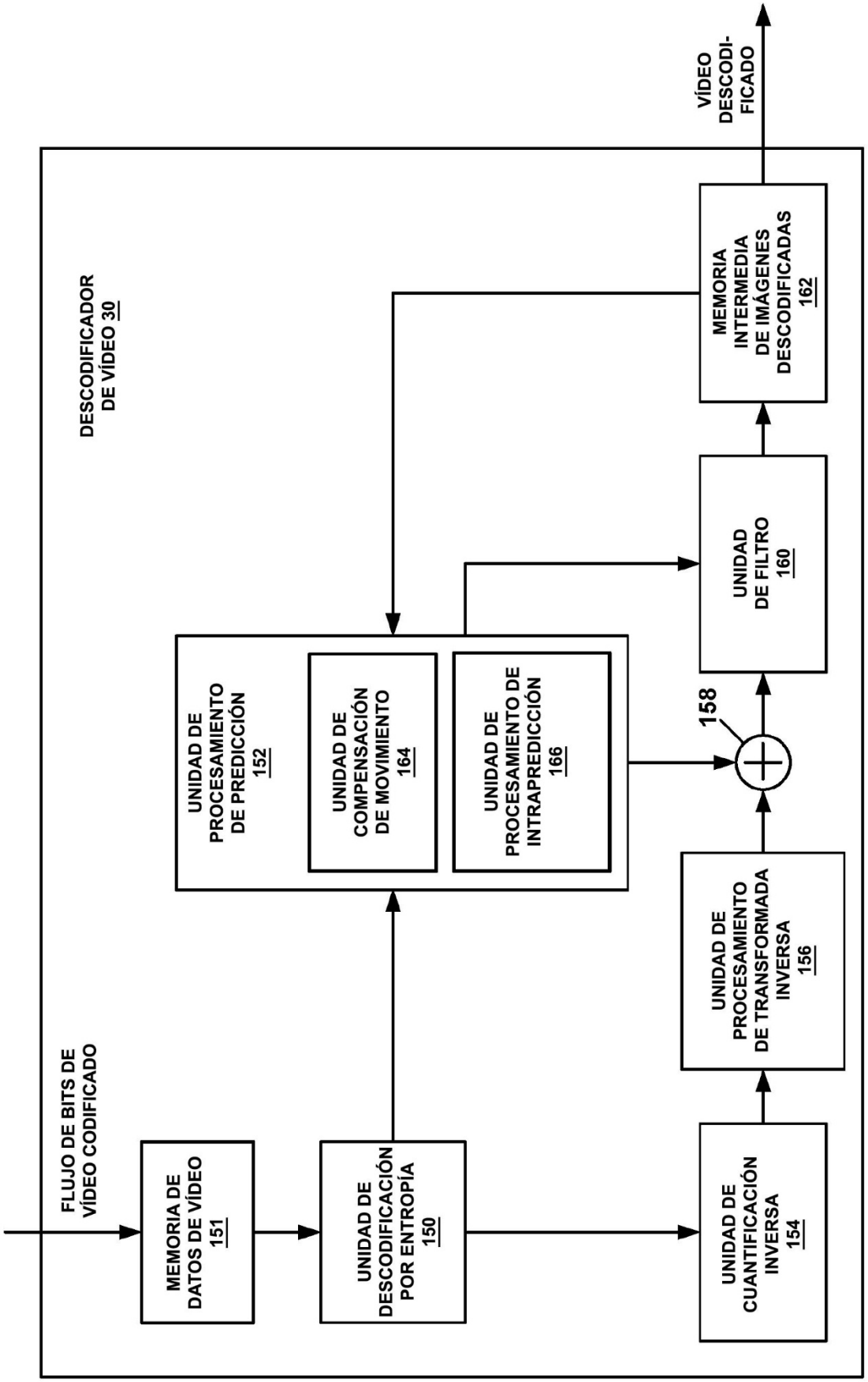
DESCODIFICADOR DE VÍDEO 30
FLUJO DE BITS DE VÍDEO CODIFICADO
MEMORIA DE DATOS DE VÍDEO 151
UNIDAD DE DESCODIFICACIÓN POR ENTROPÍA 150
UNIDAD DE PROCESAMIENTO DE PREDICCIÓN 152
UNIDAD DE COMPENSACIÓN DE MOVIMIENTO 164
UNIDAD DE PROCESAMIENTO DE INTRAPREDICCIÓN 166
UNIDAD DE CUANTIFICACIÓN INVERSA 154
UNIDAD DE PROCESAMIENTO DE TRANSFORMADA INVERSA 156
158
UNIDAD DE FILTRO 160
MEMORIA INTERMEDIA DE IMÁGENES DESCODIFICADAS 162
VÍDEO DESCODI-FICADO

FIG. 6

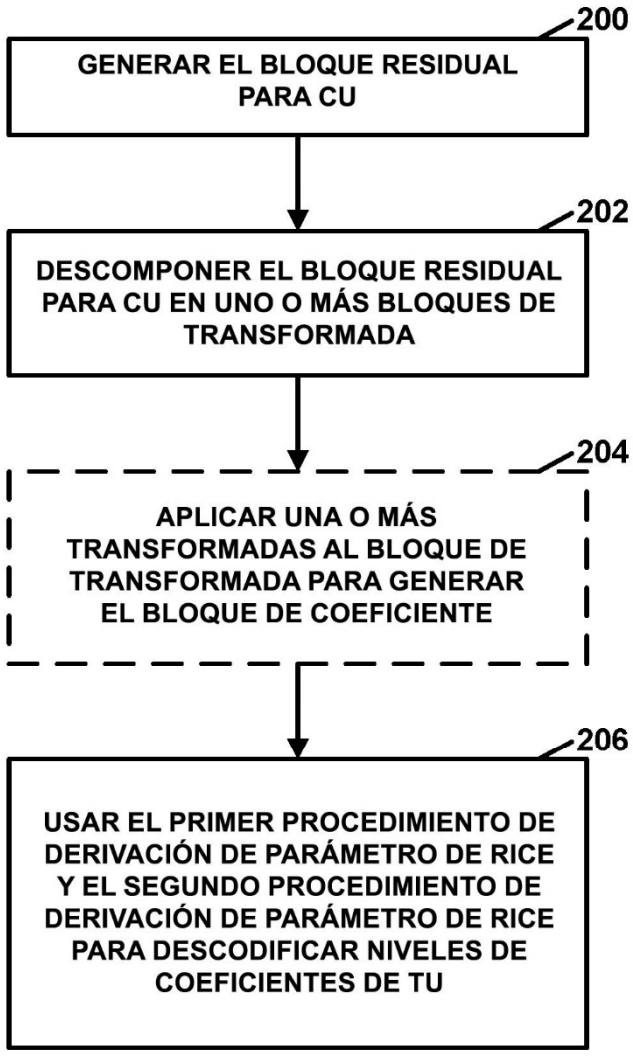
200
GENERAR EL BLOQUE RESIDUAL PARA CU
202
DESCOMPONER EL BLOQUE RESIDUAL PARA CU EN UNO O MÁS BLOQUES DE TRANSFORMADA
204
APLICAR UNA O MÁS TRANSFORMADAS AL BLOQUE DE TRANSFORMADA PARA GENERAR EL BLOQUE DE COEFICIENTE
206
USAR EL PRIMER PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE Y EL SEGUNDO PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE PARA DESCODIFICAR NIVELES DE COEFICIENTES DE TU

FIG. 7

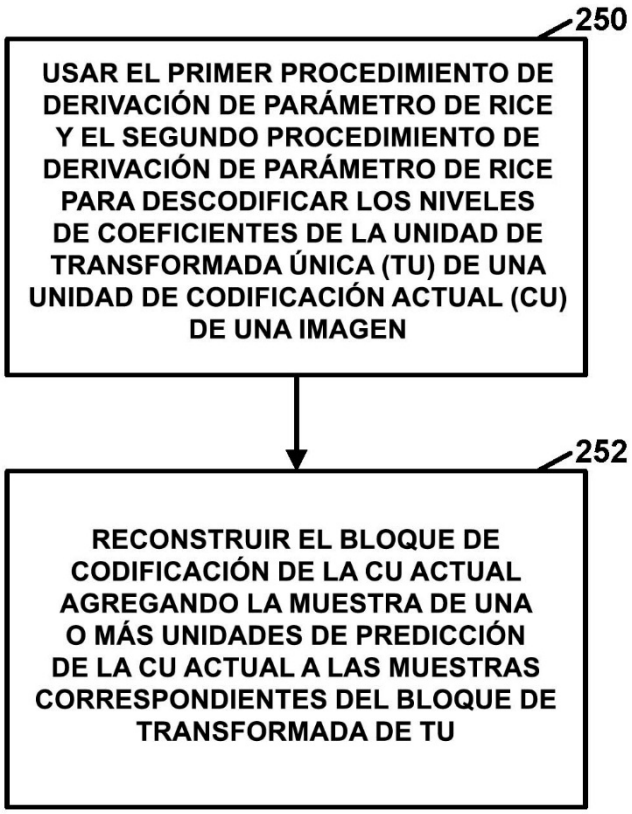
250
USAR EL PRIMER PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE Y EL SEGUNDO PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE PARA DESCODIFICAR LOS NIVELES DE COEFICIENTES DE LA UNIDAD DE TRANSFORMADA ÚNICA (TU) DE UNA UNIDAD DE CODIFICACIÓN ACTUAL (CU) DE UNA IMAGEN
252
RECONSTRUIR EL BLOQUE DE CODIFICACIÓN DE LA CU ACTUAL AGREGANDO LA MUESTRA DE UNA O MÁS UNIDADES DE PREDICCIÓN DE LA CU ACTUAL A LAS MUESTRAS CORRESPONDIENTES DEL BLOQUE DE TRANSFORMADA DE TU

FIG. 8

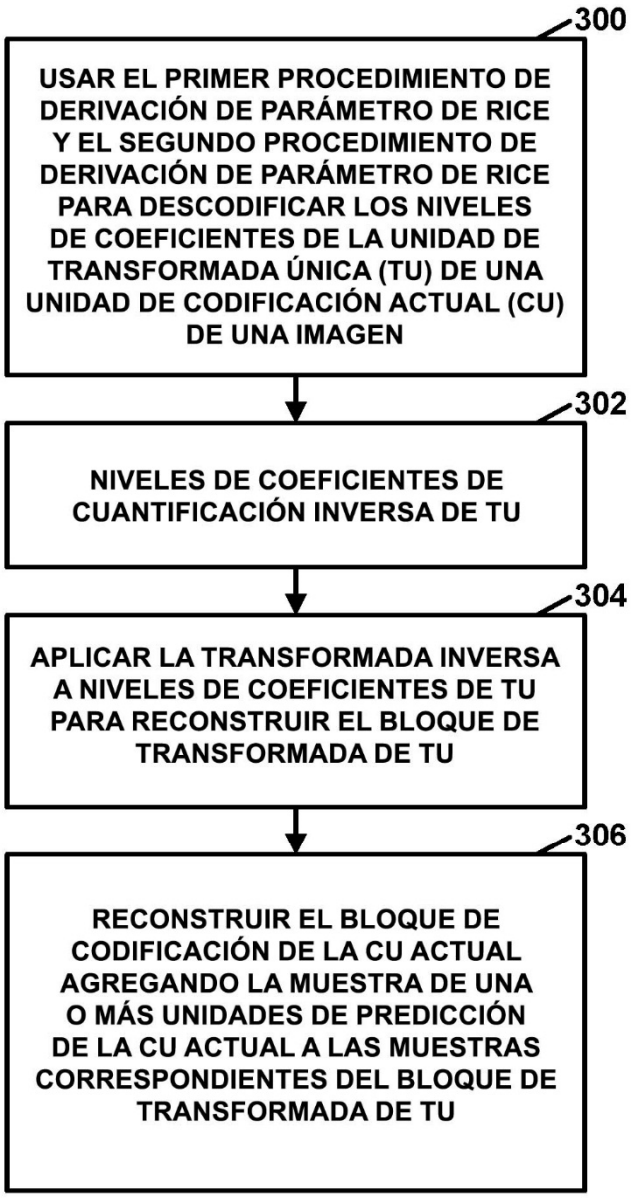
300
USAR EL PRIMER PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE Y EL SEGUNDO PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE PARA DESCODIFICAR LOS NIVELES DE COEFICIENTES DE LA UNIDAD DE TRANSFORMADA ÚNICA (TU) DE UNA UNIDAD DE CODIFICACIÓN ACTUAL (CU) DE UNA IMAGEN
302
NIVELES DE COEFICIENTES DE CUANTIFICACIÓN INVERSA DE TU
304
APLICAR LA TRANSFORMADA INVERSA A NIVELES DE COEFICIENTES DE TU PARA RECONSTRUIR EL BLOQUE DE TRANSFORMADA DE TU
306
RECONSTRUIR EL BLOQUE DE CODIFICACIÓN DE LA CU ACTUAL AGREGANDO LA MUESTRA DE UNA O MÁS UNIDADES DE PREDICCIÓN DE LA CU ACTUAL A LAS MUESTRAS CORRESPONDIENTES DEL BLOQUE DE TRANSFORMADA DE TU

FIG. 9

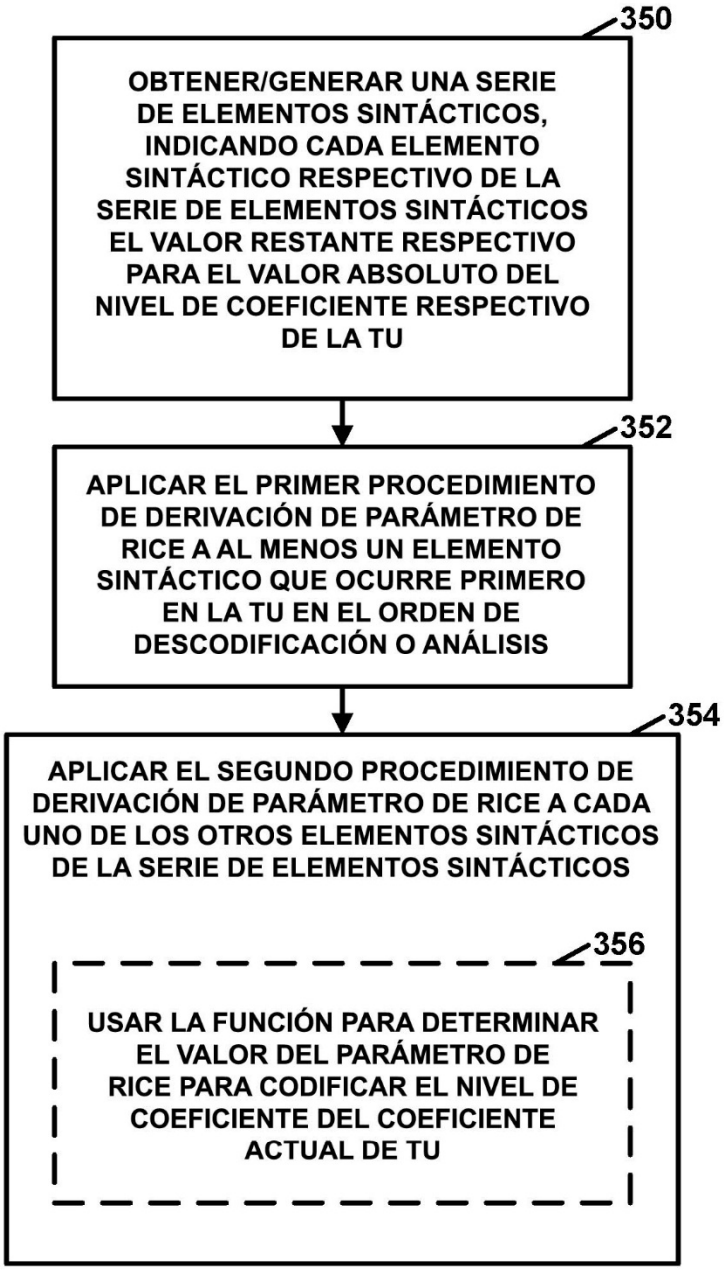
350
OBTENER/GENERAR UNA SERIE DE ELEMENTOS SINTÁCTICOS, INDICANDO CADA ELEMENTO SINTÁCTICO RESPECTIVO DE LA SERIE DE ELEMENTOS SINTÁCTICOS EL VALOR RESTANTE RESPECTIVO PARA EL VALOR ABSOLUTO DEL NIVEL DE COEFICIENTE RESPECTIVO DE LA TU
352
APLICAR EL PRIMER PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE A AL MENOS UN ELEMENTO SINTÁCTICO QUE OCURRE PRIMERO EN LA TU EN EL ORDEN DE DESCODIFICACIÓN O ANÁLISIS
354
APLICAR EL SEGUNDO PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE A CADA UNO DE LOS OTROS ELEMENTOS SINTÁCTICOS DE LA SERIE DE ELEMENTOS SINTÁCTICOS
356
USAR LA FUNCIÓN PARA DETERMINAR EL VALOR DEL PARÁMETRO DE RICE PARA CODIFICAR EL NIVEL DE COEFICIENTE DEL COEFICIENTE ACTUAL DE TU

FIG. 10

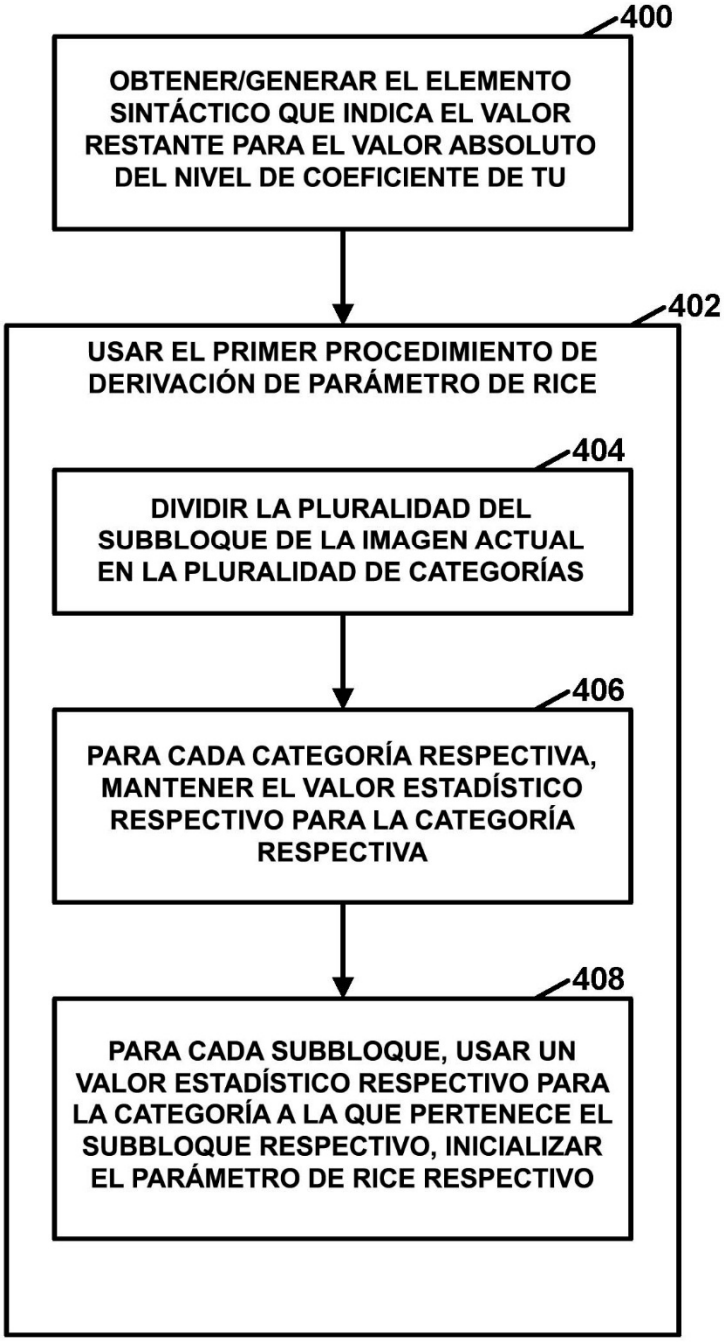
400
OBTENER/GENERAR EL ELEMENTO SINTÁCTICO QUE INDICA EL VALOR RESTANTE PARA EL VALOR ABSOLUTO DEL NIVEL DE COEFICIENTE DE TU
402
USAR EL PRIMER PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE
404
DIVIDIR LA PLURALIDAD DEL SUBBLOQUE DE LA IMAGEN ACTUAL EN LA PLURALIDAD DE CATEGORÍAS
406
PARA CADA CATEGORÍA RESPECTIVA, MANTENER EL VALOR ESTADÍSTICO RESPECTIVO PARA LA CATEGORÍA RESPECTIVA
408
PARA CADA SUBBLOQUE, USAR UN VALOR ESTADÍSTICO RESPECTIVO PARA LA CATEGORÍA A LA QUE PERTENECE EL SUBBLOQUE RESPECTIVO, INICIALIZAR EL PARÁMETRO DE RICE RESPECTIVO

FIG. 11

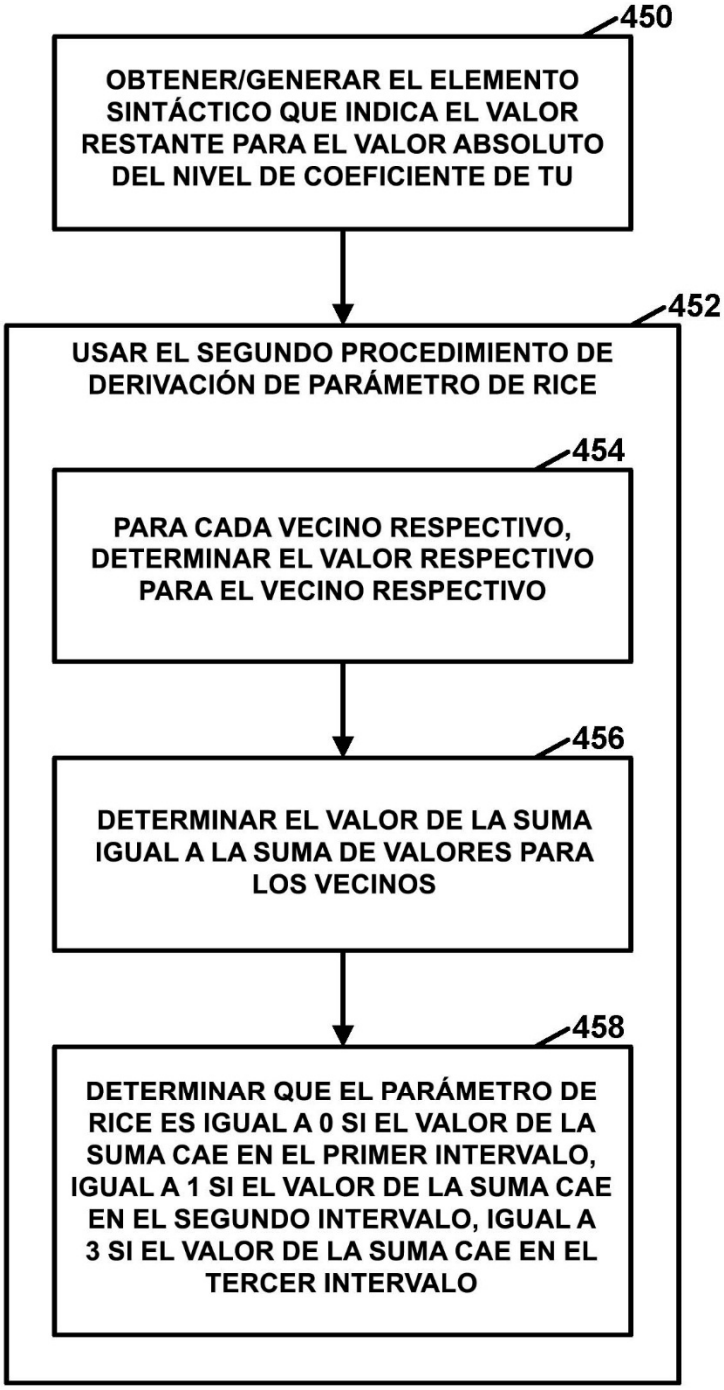
450
OBTENER/GENERAR EL ELEMENTO SINTÁCTICO QUE INDICA EL VALOR RESTANTE PARA EL VALOR ABSOLUTO DEL NIVEL DE COEFICIENTE DE TU
452
USAR EL SEGUNDO PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE
454
PARA CADA VECINO RESPECTIVO, DETERMINAR EL VALOR RESPECTIVO PARA EL VECINO RESPECTIVO
456
DETERMINAR EL VALOR DE LA SUMA IGUAL A LA SUMA DE VALORES PARA LOS VECINOS
458
DETERMINAR QUE EL PARÁMETRO DE RICE ES IGUAL A 0 SI EL VALOR DE LA SUMA CAE EN EL PRIMER INTERVALO, IGUAL A 1 SI EL VALOR DE LA SUMA CAE EN EL SEGUNDO INTERVALO, IGUAL A 3 SI EL VALOR DE LA SUMA CAE EN EL TERCER INTERVALO

FIG. 12

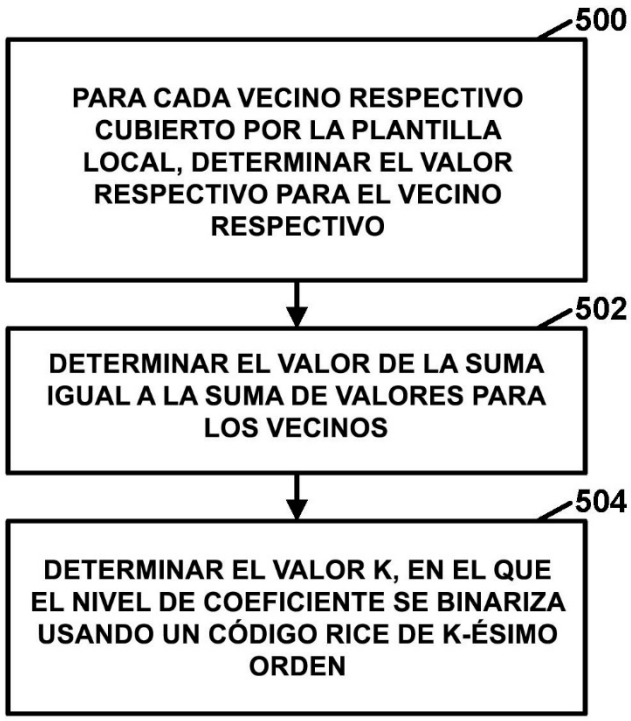
500
PARA CADA VECINO RESPECTIVO CUBIERTO POR LA PLANTILLA LOCAL, DETERMINAR EL VALOR RESPECTIVO PARA EL VECINO RESPECTIVO
502
DETERMINAR EL VALOR DE LA SUMA IGUAL A LA SUMA DE VALORES PARA LOS VECINOS
504
DETERMINAR EL VALOR K, EN EL QUE EL NIVEL DE COEFICIENTE SE BINARIZA USANDO UN CÓDIGO RICE DE K-ÉSIMO ORDEN

FIG. 13

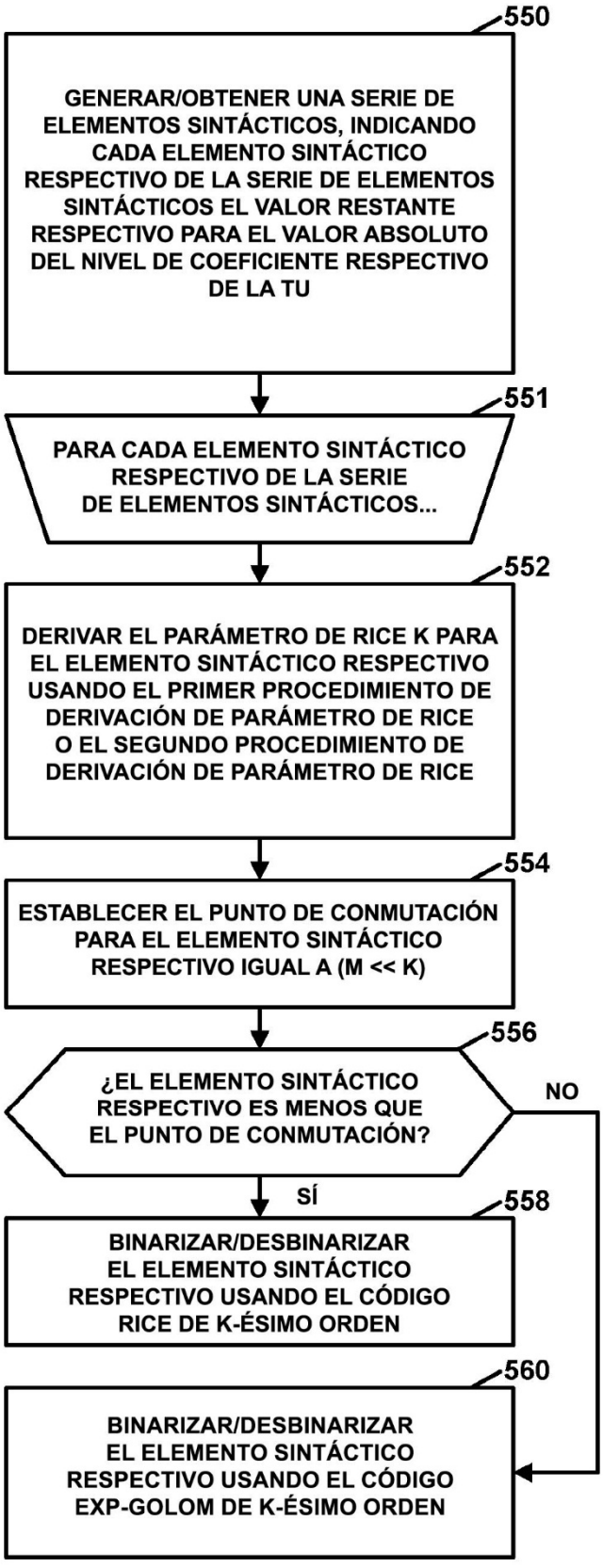
550
GENERAR/OBTENER UNA SERIE DE ELEMENTOS SINTÁCTICOS, INDICANDO CADA ELEMENTO SINTÁCTICO RESPECTIVO DE LA SERIE DE ELEMENTOS SINTÁCTICOS EL VALOR RESTANTE RESPECTIVO PARA EL VALOR ABSOLUTO DEL NIVEL DE COEFICIENTE RESPECTIVO DE LA TU
551
PARA CADA ELEMENTO SINTÁCTICO RESPECTIVO DE LA SERIE DE ELEMENTOS SINTÁCTICOS...
552
DERIVAR EL PARÁMETRO DE RICE K PARA EL ELEMENTO SINTÁCTICO RESPECTIVO USANDO EL PRIMER PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE O EL SEGUNDO PROCEDIMIENTO DE DERIVACIÓN DE PARÁMETRO DE RICE
554
ESTABLECER EL PUNTO DE CONMUTACIÓN PARA EL ELEMENTO SINTÁCTICO RESPECTIVO IGUAL A (M << K)
556
¿EL ELEMENTO SINTÁCTICO RESPECTIVO ES MENOS QUE EL PUNTO DE CONMUTACIÓN?
NO
SÍ
558
BINARIZAR/DESBINARIZAR EL ELEMENTO SINTÁCTICO RESPECTIVO USANDO EL CÓDIGO RICE DE K-ÉSIMO ORDEN
560
BINARIZAR/DESBINARIZAR EL ELEMENTO SINTÁCTICO RESPECTIVO USANDO EL CÓDIGO EXP-GOLOM DE K-ÉSIMO ORDEN

FIG. 14