(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 704 627**

(51) Int. CI.:

H04N 19/70 (2014.01)
H04N 19/13 (2014.01)
H04N 19/91 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **18.06.2012 PCT/EP2012/061613**

(87) Fecha y número de publicación internacional: **20.12.2012 WO12172113**

(96) Fecha de presentación y número de la solicitud europea: **18.06.2012 E 12734832 (4)**

(97) Fecha y número de publicación de la concesión europea: **31.10.2018 EP 2721822**

(54) Título: **Codificación entrópica de diferencias de vector de movimiento**

(30) Prioridad:

16.06.2011 US 201161497794 P
15.07.2011 US 201161508506 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
19.03.2019

(73) Titular/es:

GE VIDEO COMPRESSION, LLC (100.0%)
GE Video Compression, LLC
Albany, NY 12211, US

(72) Inventor/es:

GEORGE, VALERI;
BROSS, BENJAMIN;
KIRCHHOFFER, HEINER;
MARPE, DETLEV;
NGUYEN, TUNG;
PREISS, MATTHIAS;
SIEKMANN, MISCHA;
STEGEMANN, JAN y
WIEGAND, THOMAS

(74) Agente/Representante:

ARIZTI ACHA, Monica

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

DESCRIPCIÓN

Codificación entrópica de diferencias de vector de movimiento

La presente invención se refiere al concepto de codificación entrópica para codificar datos de vídeo.

En la técnica se conocen muchos códecs de vídeo. Generalmente, estos códecs reducen la cantidad de datos necesarios con el fin de representar el contenido de vídeo, es decir, comprimen los datos. En el contexto de la codificación de vídeo, se conoce que la compresión de los datos de vídeo se logra ventajosamente aplicando secuencialmente diferentes técnicas de codificación: se usa predicción con compensación de movimiento con el fin de predecir el contenido de imagen. Los vectores de movimiento determinados en la predicción con compensación de movimiento, así como el residuo de predicción se someten a codificación entrópica sin pérdidas. Con el fin de reducir adicionalmente la cantidad de datos, los propios vectores de movimiento se someten a predicción de modo que simplemente tienen que codificarse de manera entrópica diferencias de vector de movimiento que representan el residuo de predicción de vector de movimiento. En H.264, por ejemplo, se aplica el procedimiento que acaba de explicarse resumidamente con el fin de transmitir la información sobre diferencias de vector de movimiento. En particular, las diferencias de vector de movimiento se binarizan para dar cadenas de elementos binarios correspondientes a una combinación de un código unario truncado y, a partir de un determinado valor de corte en adelante, un código de Golomb exponencial. Mientras que los elementos binarios del código de Golomb exponencial se codifican fácilmente usando un modo de derivación de equiprobabilidad con probabilidad fijada de 0,5, se proporcionan varios contextos para los primeros elementos binarios. El valor de corte se elige para que sea nueve. Por consiguiente, se proporciona una alta cantidad de contextos para codificar las diferencias de vector de movimiento.

Sin embargo, proporcionar un alto número de contextos no sólo aumenta la complejidad de codificación, sino que también afecta negativamente a la eficiencia de codificación: si un contexto se visita con demasiado poca frecuencia, no logra realizarse eficazmente la adaptación de probabilidad, es decir la adaptación de la estimación de probabilidad asociada con el contexto respectivo durante la causa codificación entrópica. Por consiguiente, las estimaciones de probabilidad aplicadas estiman de manera inapropiada las estadísticas de símbolos reales. Además, si para un determinado elemento binario de la binarización se proporcionan varios contextos, la selección a partir de los mismos puede necesitar la inspección de valores de elementos de sintaxis/elementos binarios vecinos, necesidad que puede dificultar la ejecución del procedimiento de decodificación. Por otro lado, si el número de contextos que se proporcionan es demasiado bajo, elementos binarios con estadísticas de símbolos reales que varían sumamente se agrupan entre sí dentro de un contexto y, por consiguiente, la estimación de probabilidad asociada con ese contexto no logra codificar eficazmente los elementos binarios asociados con el mismo.

Existe una necesidad continuada de aumentar adicionalmente la eficiencia de codificación de la codificación entrópica de diferencias de vector de movimiento.

Por consiguiente, un objetivo de la presente invención es proporcionar un concepto de codificación de este tipo.

Este objetivo se logra mediante el objeto de las reivindicaciones independientes adjuntas al presente documento.

Un hallazgo básico de la presente invención es que la eficiencia de codificación de la codificación entrópica de diferencias de vector de movimiento puede aumentarse adicionalmente reduciendo el valor de corte hasta el que se usa el código unario truncado con el fin de binarizar las diferencias de vector de movimiento, hasta dos de modo que simplemente hay dos posiciones de elemento binario del código unario truncado, y si se usa un orden de uno para el código de Golomb exponencial para la binarización de las diferencias de vector de movimiento a partir el valor de corte en adelante y si, adicionalmente, se proporciona exactamente un contexto para las dos posiciones de elemento binario del código unario truncado, respectivamente, de modo que no se necesita una selección de contexto basándose en elementos binarios o valores de elementos de sintaxis de bloques de imagen vecinos y se evita una clasificación demasiado fina de los elementos binarios en esas posiciones de elemento binario en contextos de modo que la adaptación de probabilidad funciona de manera apropiada, y si se usan los mismos contextos para las componentes horizontal y vertical reduciendo así adicionalmente los efectos negativos de una subdivisión de contexto demasiado fina.

Además, se ha encontrado que los ajustes que acaban de mencionarse con respecto a la codificación entrópica de diferencias de vector de movimiento son especialmente valiosos cuando se combinan los mismos con métodos avanzados de predicción de los vectores de movimiento y reducción de la cantidad necesaria de diferencias de vector de movimiento que van a transmitirse. Por ejemplo, pueden proporcionarse múltiples factores de predicción de vector de movimiento para obtener una lista ordenada de factores de predicción de vector de movimiento, y puede usarse un índice en esta lista de factores de predicción de vector de movimiento para determinar el factor de predicción de vector de movimiento real cuyo residuo de predicción se representa por la diferencia de vector de movimiento en cuestión. Aunque la información sobre el índice de lista usado tiene que poder derivarse a partir del

flujo de datos en el lado de decodificación, se aumenta la calidad de predicción global de los vectores de movimiento y, por consiguiente, se reduce adicionalmente la magnitud de las diferencias de vector de movimiento de modo que, en conjunto, se aumenta adicionalmente la eficiencia de codificación y la reducción del valor de corte y el uso común del contexto para las componentes horizontal y vertical de las diferencias de vector de movimiento se ajusta a una predicción de vector de movimiento mejorada de este tipo. Por otro lado, puede usarse fusión con el fin de reducir el número de diferencias de vector de movimiento que van a transmitirse dentro del flujo de datos: para ello, puede transmitirse información de fusión dentro del flujo de datos indicándole al decodificador bloques de una subdivisión de bloques que se agrupan en un grupo de bloques. Después pueden transmitirse las diferencias de vector de movimiento dentro del flujo de datos en unidades de estos grupos fusionados en lugar de los bloques individuales, reduciendo así el número de diferencias de vector de movimiento que tienen que transmitirse. Dado que esta agrupación de bloques reduce la intercorrelación entre diferencias de vectores de movimiento vecinos, la omisión que acaba de mencionarse de proporcionar varios contextos para una posición de elemento binario impide que el esquema decodificación entrópica proporcione una clasificación demasiado fina en contextos dependiendo de diferencias de vectores de movimiento vecinos. En vez de eso, el concepto de fusión ya aprovecha la intercorrelación entre diferencias de vector de movimiento de bloques vecinos y, por consiguiente, un contexto para una posición de elemento binario (el mismo para las componentes horizontal y vertical) es suficiente.

El documento WIEGAND T *et al.*: “WD3: Working Draft 3 of High-Efficiency Video Coding”, 20110329, n.º JCTVC-E603, 29 de marzo de 2011 (29-03-201129), XP030009014, ISSN: 0000-0003 describe un borrador de trabajo de la norma HEVC. A continuación, se describen realizaciones preferidas de la presente solicitud con respecto a las figuras, en las que

la figura 1 muestra un diagrama de bloques de un codificador según una realización;

las figuras 2a-2c muestran esquemáticamente diferentes subdivisiones de una matriz de muestras tal como una imagen en bloques;

la figura 3 muestra un diagrama de bloques de un decodificador según una realización;

la figura 4 muestra un diagrama de bloques de un codificador según una realización con más detalle;

la figura 5 muestra un diagrama de bloques de un decodificador según una realización con más detalle;

la figura 6 ilustra esquemáticamente una transformación de un bloque del dominio espacial al dominio espectral, el bloque de transformación resultante y su retransformación;

la figura 7 muestra un diagrama de bloques de un codificador según una realización;

la figura 8 muestra un diagrama de bloques de un decodificador adecuado para decodificar el flujo de bits generado por el codificador de la figura 8, según una realización;

la figura 9 muestra un diagrama esquemático que ilustra un paquete de datos con flujos de bits parciales multiplexados según una realización;

la figura 10 muestra un diagrama esquemático que ilustra un paquete de datos con una segmentación alternativa usando segmentos de tamaño fijo según una realización adicional;

la figura 11 muestra un decodificador que soporta conmutación de modo según una realización;

la figura 12 muestra un decodificador que soporta conmutación de modo según una realización adicional;

la figura 13 muestra un codificador adaptado al decodificador de la figura 11 según una realización;

la figura 14 muestra un codificador adaptado al decodificador de la figura 12 según una realización;

la figura 15 muestra mapeo de pStateCtx y fullCtxState/256**E**;

la figura 16 muestra un decodificador según una realización de la presente invención; y

la figura 17 muestra un codificador según una realización de la presente invención;

la figura 18 muestra esquemáticamente una binarización de diferencia de vector de movimiento según una realización de la presente invención;

la figura 19 ilustra esquemáticamente un concepto de fusión según una realización; y

la figura 20 ilustra esquemáticamente un esquema de predicción de vector de movimiento según una realización.

Se indica que, durante la descripción de las figuras, elementos que se producen en varias de estas figuras se indican con el mismo signo de referencia en cada una de estas figuras y se evita una descripción repetida de estos elementos en lo que se refiere a la funcionalidad con el fin de evitar repeticiones innecesarias. No obstante, las funcionalidades y descripciones proporcionadas con respecto a una figura también se aplicarán a las otras figuras a menos que se indique explícitamente lo contrario.

A continuación, en primer lugar, se describen realizaciones de un concepto de codificación de vídeo general, con respecto a las figuras 1 a 10. Las figuras 1 a 6 se refieren a la parte del códec de vídeo que funciona a nivel de sintaxis. Las siguientes figuras 8 a 10 se refieren a realizaciones para la parte del código relacionada con la conversión del flujo de elementos de sintaxis en el flujo de datos y viceversa. Después, se describen aspectos y realizaciones específicos de la presente invención en forma de posibles implementaciones del concepto general explicado resumidamente de manera representativa con respecto a las figuras 1 a 10.

La figura 1 muestra un ejemplo para un codificador 10 en el que pueden implementarse aspectos de la presente solicitud.

El codificador codifica una matriz 20 de muestras de información para dar un flujo de datos. La matriz de muestras de información puede representar muestras de información correspondientes, por ejemplo, a valores de brillo, valores de color, valores de luma, valores de croma o similares. Sin embargo, las muestras de información también pueden ser valores de profundidad en caso de que la matriz 20 de muestras sea un mapa de profundidad generado, por ejemplo, mediante un sensor de tiempo de luz o similar.

El codificador 10 es un codificador basado en bloques. Es decir, el codificador 10 codifica la matriz 20 de muestras para dar el flujo 30 de datos en unidades de bloques 40. La codificación en unidades de bloques 40 no significa necesariamente que el codificador 10 codifica estos bloques 40 de manera totalmente independiente unos de otros. En vez de eso, el codificador 10 puede usar reconstrucciones de bloques anteriormente codificados con el fin de extrapolar o realizar intrapredicción de bloques restantes, y puede usar la glanuralidad de los bloques para establecer parámetros de codificación, es decir para establecer la manera en la que se codifica cada región de matriz de muestras correspondiente a un bloque respectivo.

Además, el codificador 10 es un codificador de transformación. Es decir, el codificador 10 codifica los bloques 40 usando una transformación con el fin de transferir las muestras de información dentro de cada bloque 40 del dominio espacial al dominio espectral. Puede usarse una transformación bidimensional tal como una DCT de FFT o similar. Preferiblemente, los bloques 40 tienen forma cuadrática o forma rectangular.

La subdivisión de la matriz 20 de muestras para dar los bloques 40 mostrada en la figura 1 simplemente sirve para fines de ilustración. La figura 1 muestra que la matriz 20 de muestras subdividiéndose en una disposición bidimensional regular de bloques 40 cuadráticos o rectangular que están en contacto unos con otros de una manera no solapante. El tamaño de los bloques 40 puede estar predeterminado. Es decir, el codificador 10 puede no transferir una información sobre el tamaño de bloque de los bloques 40 dentro del flujo 30 de datos al lado de decodificación. Por ejemplo, el decodificador puede esperar el tamaño de bloque predeterminado.

Sin embargo, son posibles varias alternativas. Por ejemplo, los bloques pueden solaparse entre sí. Sin embargo, el solapamiento puede estar restringido hasta un punto tal que cada bloque tiene una porción no solapada por ningún bloque vecino, o de tal manera que cada muestra de los bloques está solapada, como máximo, por un bloque de los bloques vecinos dispuestos en yuxtaposición con respecto al bloque actual a lo largo de una dirección predeterminada. Esto último significará que los bloques vecinos a la izquierda y a la derecha pueden solaparse con el bloque actual para cubrir completamente el bloque actual, pero pueden no solaparse entre sí, y lo mismo se aplica para los vecinos en la dirección vertical y diagonal.

Como alternativa adicional, la subdivisión de la matriz 20 de muestras para dar los bloques 40 puede adaptarse al contenido de la matriz 20 de muestras mediante el codificador 10 transfiriéndose la información de subdivisión sobre la subdivisión usada al lado de decodificador a través del flujo 30 de bits.

Las figuras 2a a 2c muestran diferentes ejemplos para una subdivisión de una matriz 20 de muestras para dar los bloques 40. La figura 2a muestra una subdivisión basada en árbol cuádruple de una matriz 20 de muestras para dar los bloques 40 de diferentes tamaños, indicándose bloques representativos en 40a, 40b, 40c y 40d con tamaño creciente. Según la subdivisión de la figura 2a, la matriz 20 de muestras se divide en primer lugar para dar una disposición bidimensional regular de los bloques 40d de árbol que, a su vez, tienen información de subdivisión individual asociada con los mismos según la cual un determinado bloque 40d de árbol puede subdividirse

adicionalmente según una estructura de árbol cuádruple o no. El bloque de árbol a la izquierda del bloque 40d se subdivide a modo de ejemplo en bloques más pequeños según una estructura de árbol cuádruple. El codificador 10 puede realizar una transformación bidimensional para cada uno de los bloques mostrados con líneas continuas y discontinuas en la figura 2a. Dicho de otro modo, el codificador 10 puede transformar la matriz 20 en unidades de la subdivisión de bloques.

En vez de una subdivisión basada en árbol cuádruple, puede usarse una subdivisión basada en árbol múltiple más general y el número de nodos secundarios por nivel de jerarquía puede diferir entre diferentes niveles de jerarquía.

La figura 2b muestra otro ejemplo para una subdivisión. Según la figura 2b, la matriz 20 de muestras se divide en primer lugar en macrobloques 40b dispuestos en una disposición bidimensional regular de una manera en contacto mutuo no solapante en la que cada macrobloque 40b tiene asociada con el mismo información de subdivisión según la cual un macrobloque no se subdivide, o, si se subdivide, se subdivide de manera bidimensional regular para dar subbloques de igual tamaño para lograr diferentes granularidades de subdivisión para diferentes macrobloques. El resultado es una subdivisión de la matriz 20 de muestras en bloques 40 de diferentes tamaños indicándose representaciones de los diferentes tamaños en 40a, 40b y 40a'. Como en la figura 2a, el codificador 10 realiza una transformación bidimensional en cada uno de los bloques mostrados en la figura 2b con las líneas continuas y discontinuas. Más adelante se comentará la figura 2c.

La figura 3 muestra un decodificador 50 que puede decodificar el flujo 30 de datos generado por el codificador 10 para reconstruir una versión 60 reconstruida de la matriz 20 de muestras. El decodificador 50 extrae del flujo 30 de datos el bloque de coeficiente de transformación para cada uno de los bloques 40 y reconstruye la versión 60 reconstruida realizando una transformación inversa en cada uno de los bloques de coeficiente de transformación.

El codificador 10 y el decodificador 50 pueden configurarse para realizar codificación/decodificación entrópica con el fin de insertar la información en los bloques de coeficiente de transformación en el, y extraer esta información del, flujo de datos, respectivamente. Más adelante se describen detalles a este respecto según diferentes realizaciones. Debe observarse que el flujo 30 de datos no comprende necesariamente información sobre bloques de coeficiente de transformación para todos los bloques 40 de la matriz 20 de muestras. En vez de eso, un subconjunto de bloques 40 pueden codificarse en el flujo 30 de bits de otra manera. Por ejemplo, el codificador 10 puede decidir abstenerse de insertar un bloque de coeficiente de transformación para un determinado bloque de los bloques 40 insertando en su lugar en el flujo 30 de bits parámetros de codificación alternativos que permiten al decodificador 50 predecir o rellenar de otro modo el bloque respectivo en la versión 60 reconstruida. Por ejemplo, el codificador 10 puede realizar un análisis de textura con el fin de localizar bloques dentro de la matriz 20 de muestras que pueden rellenarse en el lado de decodificador por el decodificador mediante síntesis de textura e indicar esto dentro del flujo de bits en consecuencia.

Tal como se comenta con respecto a las siguientes figuras, los bloques de coeficiente de transformación no representan necesariamente una representación en el dominio espectral de las muestras de información originales de un bloque 40 respectivo de la matriz 20 de muestras. En vez de eso, un bloque de coeficiente de transformación de este tipo puede representar una representación en el dominio espectral de un residuo de predicción del bloque 40 respectivo. La figura 4 muestra una realización de un codificador de este tipo. El codificador de la figura 4 comprende una etapa 100 de transformación, un codificador 102 entrópico, una etapa 104 de transformación inversa, un factor 106 de predicción y un restador 108 así como un sumador 110. El restador 108, la etapa 100 de transformación y el codificador 102 entrópico están conectados en serie en el orden mencionado entre una entrada 112 y una salida 114 del codificador de la figura 4. La etapa 104 de transformación inversa, el sumador 110 y el factor 106 de predicción están conectados en el orden mencionado entre la salida de la etapa 100 de transformación y la entrada inversa del restador 108, estando también la salida del factor 106 de predicción conectada a una entrada adicional del sumador 110.

El codificador de la figura 4 es un codificador de bloques basado en transformación predictivo. Es decir, los bloques de una matriz 20 de muestras que entran en la entrada 112 se predicen a partir de porciones anteriormente codificadas y reconstruidas de la misma matriz 20 de muestras u otras matrices de muestra anteriormente codificadas y reconstruidas que pueden preceder o suceder a la matriz 20 de muestras actual en el tiempo de presentación. La predicción se realiza mediante el factor 106 de predicción. El restador 108 resta la predicción de un bloque original de este tipo y la etapa 100 de transformación realiza una transformación bidimensional en los residuos de predicción. La propia transformación bidimensional o una medida posterior dentro de la etapa 100 de transformación puede conducir a una cuantificación de los coeficientes de transformación dentro de los bloques de coeficiente de transformación. Los bloques de coeficiente de transformación cuantificados se codifican sin pérdidas, por ejemplo, mediante codificación entrópica dentro del codificador 102 entrópico emitiéndose el flujo de datos resultante en la salida 114. La etapa 104 de transformación inversa reconstruye el residuo cuantificado y el sumador 110, a su vez, combina el residuo reconstruido con la predicción correspondiente con el fin de obtener muestras de información reconstruidas basándose en las cuales el factor 106 de predicción puede predecir los bloques de predicción actualmente codificados anteriormente mencionados. El factor 106 de predicción puede usar diferentes

modos de predicción tales como modos de intrapredicción y modos de interpredicción con el fin de predecir los bloques y los parámetros de predicción se reenvían al codificador 102 entrópico para su inserción en el flujo de datos. Para cada bloque de predicción predicho por interpredicción, se insertan datos de movimiento respectivos en el flujo de bits mediante el codificador 114 entrópico con el fin de permitir que el lado de decodificación vuelva a realizar la predicción. Los datos de movimiento para un bloque de predicción de una imagen pueden implicar una porción de sintaxis que incluye un elemento de sintaxis que representa una diferencia de vector de movimiento que codifica de manera diferencial el vector de movimiento para el bloque de predicción actual con respecto a un factor de predicción de vector de movimiento derivado, por ejemplo, mediante un método recomendado a partir de los vectores de movimiento de bloques de predicción ya codificados vecinos.

Es decir, según la realización de la figura 4, los bloques de coeficiente de transformación representan una representación espectral de un residuo de la matriz de muestras en vez de muestras de información reales de la misma. Es decir, según la realización de la figura 4, una secuencia de elementos de sintaxis puede entrar en el codificador 102 entrópico para codificarse entrópicamente para dar un flujo 114 de datos. La secuencia de elementos de sintaxis puede comprender elementos de sintaxis de diferencia de vector de movimiento para bloques de interpredicción y elementos de sintaxis que se refieren a un mapa de significación que indica posiciones de niveles de coeficiente de transformación significativos, así como elementos de sintaxis que definen los propios niveles de coeficiente de transformación significativos, para bloques de transformación.

Debe observarse que existen varias alternativas para la realización de la figura 4 habiéndose descrito algunas de ellas en la parte introductoria de la memoria descriptiva, cuya descripción se incorpora en la descripción de la figura 4 en el presente documento.

La figura 5 muestra un decodificador que puede decodificar un flujo de datos generado por el codificador de la figura 4. El decodificador de la figura 5 comprende un decodificador 150 entrópico, una etapa 152 de transformación inversa, un sumador 154 y un factor 156 de predicción. El decodificador 150 entrópico, la etapa 152 de transformación inversa y el sumador 154 están conectados en serie entre una entrada 158 y una salida 160 del decodificador de la figura 5 en el orden mencionado. Una salida adicional del decodificador 150 entrópico está conectada al factor 156 de predicción que, a su vez, está conectado entre la salida del sumador 154 y una entrada adicional del mismo. El decodificador 150 entrópico extrae, del flujo de datos que entra en el decodificador de la figura 5 en la entrada 158, los bloques de coeficiente de transformación en los que se aplica una transformación inversa a los bloques de coeficiente de transformación en la etapa 152 con el fin de obtener la señal residual. La señal residual se combina con una predicción del factor 156 de predicción en el sumador 154 para obtener un bloque reconstruido de la versión reconstruida de la matriz de muestras en la salida 160. Basándose en las versiones reconstruidas, el factor 156 de predicción genera las predicciones reconstruyendo así las predicciones realizadas mediante el factor 106 de predicción en el lado de codificador. Con el fin de obtener las mismas predicciones que las usadas en el lado de codificador, el factor 156 de predicción usa los parámetros de predicción que el decodificador 150 entrópico también obtiene del flujo de datos en la entrada 158.

Debe observarse que, en las realizaciones descritas anteriormente, la glanuralidad espacial a la que se realiza la predicción y la transformación del residuo, no necesitan ser iguales entre sí. Esto se muestra en la figura 2C. Esta figura muestra una subdivisión para los bloques de predicción de la glanuralidad de predicción con líneas continuas y la glanuralidad de residuo con líneas discontinuas. Tal como puede observarse, las subdivisiones pueden seleccionarse por el codificador de manera independiente una de otra. Más precisamente, la sintaxis de flujo de datos puede permitir una definición de la subdivisión de residuo independiente de la subdivisión de predicción. Alternativamente, la subdivisión de residuo puede ser una extensión de la subdivisión de predicción de modo que cada bloque de residuo es o bien igual a, o bien un subconjunto apropiado de, un bloque de predicción. Esto se muestra en la figura 2a y la figura 2b, por ejemplo, en las que de nuevo la glanuralidad de predicción se muestra con líneas continuas y la glanuralidad de residuo con líneas discontinuas. Es decir, en las figuras 2a-2c, todos los bloques que tienen un signo de referencia asociado con los mismos serán bloques de residuo para los que se realizará una transformación bidimensional mientras que los bloques con bloques con línea continua más grandes que abarcan los bloques 40a con línea discontinua, por ejemplo, serán bloques de predicción para los que se realiza individualmente un ajuste de parámetro de predicción.

Las realizaciones anteriores tienen en común que un bloque de muestras (residuales u originales) va a transformarse en el lado de codificador en un bloque de coeficiente de transformación que, a su vez, va a someterse a transformación inversa para dar un bloque reconstruido de muestras en el lado de decodificador. Esto se ilustra en la figura 6. La figura 6 muestra un bloque 200 de muestras. En el caso de la figura 6, este bloque 200 es, a modo de ejemplo, cuadrático y tiene un tamaño de 4x4 muestras 202. Las muestras 202 están dispuestas de manera regular a lo largo de una dirección horizontal x y dirección vertical y. Mediante la transformación bidimensional T anteriormente mencionada, se transforma el bloque 200 en el dominio espectral, concretamente en un bloque 204 de coeficientes 206 de transformación, teniendo el bloque 204 de transformación el mismo tamaño que el bloque 200. Es decir, el bloque 204 de transformación tiene tantos coeficientes 206 de transformación como muestras tiene el bloque 200, tanto en la dirección horizontal como en la dirección vertical. Sin embargo, dado que la transformación

T es una transformación espectral, las posiciones de los coeficientes 206 de transformación dentro del bloque 204 de transformación no corresponden a las posiciones espaciales sino más bien a las componentes espectrales del contenido del bloque 200. En particular, el eje horizontal del bloque 204 de transformación corresponde a un eje a lo largo del cual la frecuencia espectral en la dirección horizontal aumenta de manera monotónica mientras que el eje vertical corresponde a un eje a lo largo del cual la frecuencia espacial en la dirección vertical aumenta de manera monotónica en el que el coeficiente de transformación de componente de DC está situado en una esquina (en este caso, a modo de ejemplo, la esquina superior izquierda) del bloque 204 de modo que en la esquina inferior derecha está situado el coeficiente 206 de transformación correspondiente a la mayor frecuencia en la dirección tanto horizontal como vertical. Despreciando la dirección espacial, la frecuencia espacial a la que pertenece un determinado coeficiente 206 de transformación aumenta generalmente desde la esquina superior izquierda hasta la esquina inferior derecha. Mediante una transformación inversa T^{-1}, el bloque 204 de transformación se transfiere de vuelta desde el dominio espectral hasta el dominio espacial, para volver a obtener una copia 208 del bloque 200. En caso de no haberse introducido ninguna cuantificación/pérdida durante la transformación, la reconstrucción será perfecta.

Tal como ya se indicó anteriormente, puede observarse a partir de la figura 6 que tamaños de bloque superiores del bloque 200 aumentan la resolución espectral de la representación 204 espectral resultante. Por otro lado, el ruido de cuantificación tiende a extenderse sobre todo el bloque 208 y, por tanto, objetos abruptos y muy localizados dentro de los bloques 200 tienden a conducir a desviaciones del bloque retransformado con respecto al bloque 200 original debido al ruido de cuantificación. Sin embargo, la principal ventaja de usar bloques más grandes es que la razón entre el número de coeficientes de transformación (cuantificados) significativos, es decir distintos de cero, es decir niveles, por un lado y el número de coeficientes de transformación insignificantes por otro lado pueden reducirse dentro de bloques más grandes en comparación con bloques más pequeños, permitiendo así una mejor eficiencia de codificación. Dicho de otro modo, con frecuencia, los niveles de coeficiente de transformación significativos, es decir los coeficientes de transformación no cuantificados a cero, se distribuyen por el bloque 204 de transformación de manera dispersa. Debido a esto, según las realizaciones descritas con más detalle a continuación, las posiciones de los niveles de coeficiente de transformación significativos se indican dentro del flujo de datos mediante un mapa de significación. De manera independiente de lo mismo, los valores de los coeficientes de transformación significativos, es decir, los niveles de coeficiente de transformación en el caso de que los coeficientes de transformación estén cuantificados, se transmiten dentro del flujo de datos.

Por tanto, todos los codificadores y decodificadores descritos anteriormente están configurados para tratar con una cierta sintaxis de elementos de sintaxis. Es decir, se supone que los elementos de sintaxis anteriormente mencionados tales como los niveles de coeficiente de transformación, elementos de sintaxis referentes al mapa de significación de bloques de transformación, los elementos de sintaxis de datos de movimiento referentes a bloques de interpredicción y así sucesivamente, están dispuestos de manera secuencial dentro del flujo de datos de una manera recomendada. Tal manera recomendada puede representarse en forma de un pseudocódigo tal como se hace, por ejemplo, en la norma H.264 u otros códecs de vídeo.

Dicho incluso de otro modo, la descripción anterior trataba principalmente con la conversión de datos de medios, en este caso a modo de ejemplo datos de vídeo, en una secuencia de elementos de sintaxis según una estructura de sintaxis predefinida que recomienda determinados tipos de elementos de sintaxis, su semántica y el orden entre los mismos. El codificador entrópico y decodificador entrópico de las figuras 4 y 5 pueden estar configurados para funcionar, y pueden estar estructurados, tal como se explica resumidamente a continuación. Los mismos son responsables de realizar la conversión entre la secuencia de elementos de sintaxis y el flujo de datos, es decir flujo de bits o símbolos.

En la figura 7 se ilustra un codificador entrópico según una realización. El codificador convierte sin pérdidas un flujo de elementos 301 de sintaxis en un conjunto de dos o más flujos 312 de bits parciales.

En una realización preferida de la invención, cada elemento 301 de sintaxis se asocia con una categoría de un conjunto de una o más categorías, es decir un tipo de elemento de sintaxis. Como ejemplo, las categorías pueden especificar el tipo del elemento de sintaxis. En el contexto de codificación de vídeo híbrida, puede asociarse una categoría independiente con modos de codificación de macrobloques, modos de codificación de bloques, índices de imagen de referencia, diferencias de vector de movimiento, indicadores de subdivisión, indicadores de bloque codificado, parámetros de cuantificación, niveles de coeficiente de transformación, etc. En otros campos de aplicación tales como codificación de audio, voz, texto, documentos o datos en general, son posibles diferentes clasificaciones de elementos de sintaxis.

En general, cada elemento de sintaxis puede adoptar un valor de un conjunto finito o infinito contable de valores, en el que el conjunto de valores de elementos de sintaxis posibles puede diferir para diferentes categorías de elementos de sintaxis. Por ejemplo, hay elementos de sintaxis binarios, así como otros con valores de números enteros.

Para reducir la complejidad del algoritmo de codificación y decodificación y para permitir un diseño de codificación y decodificación general para diferentes elementos de sintaxis y categorías de elementos de sintaxis, los elementos 301 de sintaxis se convierten en conjuntos ordenados de decisiones binarias y después se procesan esas decisiones binarias mediante algoritmos de codificación binaria simples. Por tanto, el binarizador 302 mapea de manera biyectiva el valor de cada elemento 301 de sintaxis en una secuencia (o cadena o palabra) de elementos 303 binarios. La secuencia de elementos 303 binarios representa un conjunto de decisiones binarias ordenadas. Cada elemento 303 binario o decisión binaria puede adoptar un valor de un conjunto de dos valores, por ejemplo, uno de los valores 0 y 1. El esquema de binarización puede ser diferente para diferentes categorías de elementos de sintaxis. El esquema de binarización para una categoría de elementos de sintaxis particular puede depender del conjunto de valores de elementos de sintaxis posibles y/u otras propiedades del elemento de sintaxis para la categoría particular.

La tabla 1 ilustra tres esquemas de binarización a modo de ejemplo para conjuntos infinitos contables. Los esquemas de binarización para conjuntos infinitos contables también pueden aplicarse para conjuntos finitos de valores de elementos de sintaxis. En particular para grandes conjuntos finitos de valores de elementos de sintaxis, la ineficiencia (resultante de secuencias de elementos binarios sin usar) puede ser despreciable, pero la universalidad de tales esquemas de binarización proporciona una ventaja en cuanto a complejidad y requisitos de memoria. Para conjuntos finitos pequeños de valores de elementos de sintaxis, con frecuencia es preferible (en cuanto a eficiencia de codificación) adaptar el esquema de binarización al número de valores de símbolo posibles.

La tabla 2 ilustra tres esquemas de binarización a modo de ejemplo para conjuntos finitos de 8 valores. Los esquemas de binarización para conjuntos finitos pueden derivarse a partir de los esquemas de binarización universales para conjuntos infinitos contables modificando algunas secuencias de elementos binarios de manera que los conjuntos finitos de secuencias de elementos binarios representan un código libre de redundancia (y posiblemente reordenando las secuencias de elementos binarios). Como ejemplo, el esquema de binarización unario truncado en la tabla 2 se creó modificando la secuencia de elementos binarios para el elemento de sintaxis 7 de la binarización unaria universal (véase la tabla 1). La binarización de Golomb exponencial truncada y reordenada de orden 0 en la tabla 2 se creó modificando la secuencia de elementos binarios para el elemento de sintaxis 7 de la binarización de orden 0 de Golomb exponencial universal (véase la tabla 1) y reordenando las secuencias de elementos binarios (la secuencia de elementos binarios truncada para el símbolo 7 se asignó al símbolo 1). Para conjuntos finitos de elementos de sintaxis, también es posible usar esquemas de binarización no sistemáticos / no universales, tal como se muestra a modo de ejemplo en la última columna de la tabla 2.

Tabla 1: Ejemplos de binarización para conjuntos infinitos contables (o grandes conjuntos finitos).

Valor de símbolo	Binarización unaria	Binarización de orden 0 de Golomb exponencial	Binarización de orden 1 de Golomb exponencial
0	1	1	10
1	01	010	11
2	001	011	0100
3	0001	0010 0	0101
4	0000 1	0010 1	0110
5	0000 01	0011 0	0111
6	0000 001	0011 1	0010 00
7	0000 0001	0001 000	0010 01
...	...	...	...

Tabla 2: Ejemplos de binarización para conjuntos finitos.

Valor de símbolo	binarización unaria truncada	Binarización de orden 0 de Golomb exponencial truncada y reordenada	Binarización no sistemática
0	1	1	000
1	01	000	001
2	001	010	01
3	0001	011	1000
4	0000 1	0010 0	1001
5	0000 01	0010 1	1010
6	0000 001	0011 0	1011 0
7	0000 000	0011 1	1011 1

Cada elemento 303 binario de la secuencia de elementos binarios creada mediante el binarizador 302 se alimenta al asignador 304 de parámetros en orden secuencial. El asignador de parámetros asigna un conjunto de uno o más parámetros a cada elemento 303 binario y emite el elemento binario con el conjunto asociado de parámetros 305. El

conjunto de parámetros se determina exactamente de la misma manera en el codificador y en el decodificador. El conjunto de parámetros puede consistir en uno o más de los siguientes parámetros:

En particular, el asignador 304 de parámetros puede estar configurado para asignar a un elemento 303 binario actual un modelo de contexto. Por ejemplo, el asignador 304 de parámetros puede seleccionar uno de los índices de contexto disponibles para el elemento 303 binario actual. El conjunto disponible de contextos para un elemento 303 binario actual puede depender del tipo del elemento binario que, a su vez, puede estar definido por el tipo/categoría del elemento 301 de sintaxis, la binarización de la que forma parte el elemento 303 binario actual, y una posición del elemento 303 binario actual dentro de esta última binarización. La selección de contexto entre el conjunto de contextos disponibles puede depender de elementos binarios anteriores y los elementos de sintaxis asociados con estos últimos. Cada uno de estos contextos tiene un modelo de probabilidad asociado con el mismo, es decir una medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual. El modelo de probabilidad puede ser en particular una medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario actual, definiéndose adicionalmente un modelo de probabilidad mediante un identificador que especifica una estimación de cuál de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable o más probable para el elemento 303 binario actual. En el caso de estar disponible simplemente un contexto para el elemento binario actual, la selección de contexto puede omitirse. Tal como se explicará resumidamente con más detalle a continuación, el asignador 304 de parámetros también puede realizar una adaptación de modelo de probabilidad con el fin de adaptar los modelos de probabilidad asociados con los diversos contextos a las estadísticas de elementos binarios reales de los elementos binarios respectivos que pertenecen a los contextos respectivos.

Tal como también se describirá con más detalle a continuación, el asignador 304 de parámetros puede funcionar de manera diferente dependiendo de que se active un modo de alta eficiencia (HE) o modo de baja complejidad (LC). En ambos modos el modelo de probabilidad asocia el elemento 303 binario actual a cualquiera de los codificadores 310 de elementos binarios tal como se explicará resumidamente a continuación, pero el modo de funcionamiento del asignador 304 de parámetros tiende a ser menos complejo en el modo de LC, sin embargo aumentándose la eficiencia de codificación en el modo de alta eficiencia debido a que el asignador 304 de parámetros hace que la asociación de los elementos 303 binarios individuales con los codificadores 310 individuales se adapte de manera más precisa a las estadísticas de elementos binarios, optimizando así la entropía con respecto al modo de LC.

Cada elemento binario con un conjunto de parámetros 305 asociado que se emite del asignador 304 de parámetros se alimenta a un selector 306 de memoria intermedia de elementos binarios. El selector 306 de memoria intermedia de elementos binarios modifica posiblemente el valor del elemento 305 binario de entrada basándose en el valor de elemento binario de entrada y los parámetros 305 asociados y alimenta el elemento 307 binario de salida (con un valor posiblemente modificado) a una de dos o más memorias 308 intermedias de elementos binarios. La memoria 308 intermedia de elementos binarios a la que se envía el elemento 307 binario de salida se determina basándose en el valor del elemento 305 binario de entrada y/o el valor de los parámetros 305 asociados.

En una realización preferida de la invención, el selector 306 de memoria intermedia de elementos binarios no modifica el valor del elemento binario, es decir, el elemento 307 binario de salida siempre tiene el mismo valor que el elemento 305 binario de entrada. En una realización preferida adicional de la invención, el selector 306 de memoria intermedia de elementos binarios determina el valor 307 de elemento binario de salida basándose en el valor 305 de elemento binario de entrada y la medida asociada de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual. En una realización preferida de la invención, el valor 307 de elemento binario de salida se establece igual al valor 305 de elemento binario de entrada si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual es menor que (o menor que o igual a) un umbral particular; si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual es mayor que o igual a (o mayor que) un umbral particular, el valor 307 de elemento binario de salida se modifica (es decir, se establece al opuesto del valor de elemento binario de entrada). En una realización preferida adicional de la invención, el valor 307 de elemento binario de salida se establece igual al valor 305 de elemento binario de entrada si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual es mayor que (o mayor que o igual a) un umbral particular; si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual es menor que o igual a (o menor que) un umbral particular, el valor 307 de elemento binario de salida se modifica (es decir, se establece al opuesto del valor de elemento binario de entrada). En una realización preferida de la invención, el valor del umbral corresponde a un valor de 0,5 para la probabilidad estimada para ambos valores de elemento binario posibles.

En una realización preferida adicional de la invención, el selector 306 de memoria intermedia de elementos binarios determina el valor 307 de elemento binario de salida basándose en el valor 305 de elemento binario de entrada y el identificador asociado que especifica una estimación de cuál de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable o más probable para el elemento binario actual. En una

realización preferida de la invención, el valor 307 de elemento binario de salida se establece igual al valor 305 de elemento binario de entrada si el identificador especifica que el primero de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable (o más probable) para el elemento binario actual, y el valor 307 de elemento binario de salida se modifica (es decir, se establece al opuesto del valor de elemento binario de entrada) si el identificador especifica que el segundo de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable (o más probable) para el elemento binario actual.

En una realización preferida de la invención, el selector 306 de memoria intermedia de elementos binarios determina la memoria 308 intermedia de elementos binarios a la que se envía el elemento 307 binario de salida basándose en la medida asociada de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual. En una realización preferida de la invención, el conjunto de valores posibles para la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles es finito y el selector 306 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 308 intermedia de elementos binarios con cada valor posible para la estimación de la probabilidad para uno de los dos valores de elemento binario posibles, en el que diferentes valores para la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles pueden asociarse con la misma memoria 308 intermedia de elementos binarios. En una realización preferida adicional de la invención, el intervalo de valores posibles para la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles se reparte en varios intervalos, el selector 306 de memoria intermedia de elementos binarios determina el índice de intervalo para la medida actual de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles, y el selector 306 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 308 intermedia de elementos binarios con cada valor posible para el índice de intervalo, en el que diferentes valores para el índice de intervalo pueden asociarse con la misma memoria 308 intermedia de elementos binarios. En una realización preferida de la invención, elementos 305 binarios de entrada con medidas opuestas de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles (medidas opuestas son aquellas que representan estimaciones de probabilidad P y 1 - P) se alimentan a la misma memoria 308 intermedia de elementos binarios. En una realización preferida adicional de la invención, la asociación de la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario actual con una memoria intermedia de elementos binarios particular se adapta a lo largo del tiempo, por ejemplo, con el fin de garantizar que los flujos de bits parciales creados tienen tasas de transferencia de bits similares. Más a continuación, el índice de intervalo también se denominará índice de PIPE, mientras que el índice de PIPE junto con un índice de refinamiento y un indicador que indica el valor de elemento binario más probable indexa el modelo de probabilidad real, es decir, la estimación de probabilidad.

En una realización preferida adicional de la invención, el selector 306 de memoria intermedia de elementos binarios determina la memoria 308 intermedia de elementos binarios a la que se envía el elemento 307 binario de salida basándose en la medida asociada de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario actual. En una realización preferida de la invención, el conjunto de valores posibles para la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable es finito y el selector 306 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 308 intermedia de elementos binarios con cada valor posible de la estimación de la probabilidad para el valor de elemento binario menos probable o más probable, en el que diferentes valores para la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable pueden asociarse con la misma memoria 308 intermedia de elementos binarios. En una realización preferida adicional de la invención, el intervalo de valores posibles para la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable se reparte en varios intervalos, el selector 306 de memoria intermedia de elementos binarios determina el índice de intervalo para la medida actual de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable, y el selector 306 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 308 intermedia de elementos binarios con cada valor posible para el índice de intervalo, en el que diferentes valores para el índice de intervalo pueden asociarse con la misma memoria 308 intermedia de elementos binarios. En una realización preferida adicional de la invención, la asociación de la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario actual con una memoria intermedia de elementos binarios particular se adapta a lo largo del tiempo, por ejemplo, con el fin de garantizar que el flujos de bits parciales creados tienen tasas de transferencia de bits similares.

Cada una de las dos o más memorias 308 intermedias de elementos binarios está conectada exactamente con un codificador 310 de elementos binarios y cada codificador de elementos binarios solo está conectado con una memoria 308 intermedia de elementos binarios. Cada codificador 310 de elementos binarios lee elementos binarios a partir de la memoria 308 intermedia de elementos binarios asociada y convierte una secuencia de elementos 309 binarios en una palabra 311 de código, que representa una secuencia de bits. Las memorias 308 intermedias de elementos binarios representan memorias intermedias de primero en entrar primero en salir; los elementos binarios que se alimentan más tarde (en orden secuencial) en una memoria 308 intermedia de elementos binarios no se codifican antes que elementos binarios que se alimentan más temprano (en orden secuencial) en la memoria

intermedia de elementos binarios. Las palabras 311 de código que se emiten de un codificador 310 de elementos binarios particular se escriben en un flujo 312 de bits parcial particular. El algoritmo de codificación global convierte los elementos 301 de sintaxis en dos o más flujos 312 de bits parciales, en el que el número de flujos de bits parciales es igual al número de memorias intermedias de elementos binarios y codificadores de elementos binarios. En una realización preferida de la invención, un codificador 310 de elementos binarios convierte un número variable de elementos 309 binarios en una palabra 311 de código de un número variable de bits. Una ventaja de las realizaciones de la invención explicadas de manera resumida anteriormente y a continuación es que la codificación de elementos binarios puede realizarse en paralelo (por ejemplo, para diferentes grupos de medidas de probabilidad), lo que reduce el tiempo de procesamiento para varias implementaciones.

Otra ventaja de realizaciones de la invención es que la codificación de elementos binarios, que se realiza mediante los codificadores 310 de elementos binarios, puede diseñarse específicamente para diferentes conjuntos de parámetros 305. En particular, la codificación y codificación de elementos binarios pueden optimizarse (en cuanto a eficiencia de codificación y/o complejidad) para diferentes grupos de probabilidades estimadas. Por un lado, esto permite una reducción de la complejidad de codificación/decodificación, y, por otro lado, permite una mejora de la eficiencia de codificación. En una realización preferida de la invención, los codificadores 310 de elementos binarios implementan diferentes algoritmos de codificación (es decir mapeo de secuencias de elementos binarios en palabras de código) para diferentes grupos de medidas de una estimación de la probabilidad para uno de los dos valores 305 de elemento binario posibles para el elemento binario actual. En una realización preferida adicional de la invención, los codificadores 310 de elementos binarios implementan diferentes algoritmos de codificación para diferentes grupos de medidas de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario actual.

En una realización preferida de la invención, los codificadores 310 de elementos binarios (o uno o más de los codificadores de elementos binarios) representan codificadores entrópicos que mapean directamente secuencias de elementos 309 binarios de entrada en palabras 310 de código. Tales mapeos pueden implementarse de manera eficiente y no requieren un motor de codificación aritmética complejo. El mapeo inverso de palabras de código en secuencias de elementos binarios (tal como se realiza en el decodificador) debe ser único con el fin de garantizar una decodificación perfecta de la secuencia de entrada, pero el mapeo de secuencias de elementos 309 binarios en palabras 310 de código no necesita ser necesariamente único, es decir, es posible que una secuencia de elementos binarios particular pueda mapearse en más de una secuencia de palabras de código. En una realización preferida de la invención, el mapeo de secuencias de elementos 309 binarios de entrada en palabras 310 de código es biyectivo. En una realización preferida adicional de la invención, los codificadores 310 de elementos binarios (o uno o más de los codificadores de elementos binarios) representan codificadores entrópicos que mapean directamente secuencias de longitud variable de elementos 309 binarios de entrada en palabras 310 de código de longitud variable. En una realización preferida de la invención, las palabras de código de salida representan códigos libres de redundancia tales como códigos de Huffman generales o códigos de Huffman canónicos.

En la tabla 3 se ilustran dos ejemplos para el mapeo biyectivo de secuencias de elementos binarios en códigos libres de redundancia. En una realización preferida adicional de la invención, las palabras de código de salida representan códigos redundantes adecuados para la detección de errores y recuperación de errores. En una realización preferida adicional de la invención, las palabras de código de salida representan códigos de cifrado adecuados para cifrar los elementos de sintaxis.

Tabla 3: Ejemplos de mapeos entre secuencias de elementos binarios y palabras de código.

Secuencia de elementos binarios (el orden de elementos binarios es de izquierda a derecha)	Palabras de código (el orden de bits es de izquierda a derecha)
0000 0000 0000 0001 0000 001 0000 01 0000 1 0001 001 01 1	1 0000 0001 0010 0011 0100 0101 0110 0111
000 01 001 11 1000 0 1001 1010	10 11 010 011 0001 0010 0011

1000 1	0000 0
1011	0000 1

En una realización preferida adicional de la invención, los codificadores 310 de elementos binarios (o uno o más de los codificadores de elementos binarios) representan codificadores entrópicos que mapean directamente secuencias de longitud variable de elementos 309 binarios de entrada en palabras 310 de código de longitud fija. En una realización preferida adicional de la invención, los codificadores 310 de elementos binarios (o uno o más de los codificadores de elementos binarios) representan codificadores entrópicos que mapean directamente secuencias de longitud fija de elementos 309 binarios de entrada en palabras 310 de código de longitud variable.

El decodificador según una realización de la invención se ilustra en la figura 8. El decodificador realiza básicamente las operaciones inversas al codificador, de modo que la secuencia (anteriormente codificada) de elementos 327 de sintaxis se decodifica a partir de un conjunto de dos o más flujos 324 de bits parciales. El decodificador incluye dos flujos de proceso diferentes: un flujo para solicitudes de datos, que copia el flujo de datos del codificador, y un flujo de datos, que representa la inversa del flujo de datos del codificador. En la ilustración en la figura 8, las flechas discontinuas representan el flujo de solicitud de datos, mientras que las flechas continuas representan el flujo de datos. Los bloques estructurales del decodificador básicamente copian los bloques estructurales del codificador, pero implementan las operaciones inversas.

La decodificación de un elemento de sintaxis se activa mediante una solicitud de un nuevo elemento 313 de sintaxis decodificado que se envía al binarizador 314. En una realización preferida de la invención, cada solicitud de un nuevo elemento 313 de sintaxis decodificado se asocia con una categoría de un conjunto de una o más categorías. La categoría que se asocia con una solicitud de un elemento de sintaxis es la misma que la categoría que se asoció con el elemento de sintaxis correspondiente durante la codificación.

El binarizador 314 mapea la solicitud de un elemento 313 de sintaxis en una o más solicitudes de un elemento binario que se envían al asignador 316 de parámetros. Como respuesta final a una solicitud de un elemento binario que se envía al asignador 316 de parámetros por el binarizador 314, el binarizador 314 recibe un elemento 326 binario decodificado desde el selector 318 de memoria intermedia de elementos binarios. El binarizador 314 compara la secuencia recibida de elementos 326 binarios decodificados con las secuencias de elementos binarios de un esquema de binarización particular para el elemento de sintaxis solicitado y, si la secuencia recibida de elementos 26 binarios decodificados coincide con la binarización de un elemento de sintaxis, el binarizador vacía su memoria intermedia de elementos binarios y emite el elemento de sintaxis decodificado como respuesta final a la solicitud de un nuevo símbolo decodificado. Si la secuencia ya recibida de elementos binarios decodificados no coincide con ninguna de las secuencias de elementos binarios para el esquema de binarización para el elemento de sintaxis solicitado, el binarizador envía otra solicitud de un elemento binario al asignador de parámetros hasta que la secuencia de elementos binarios decodificados coincide con una de las secuencias de elementos binarios del esquema de binarización para el elemento de sintaxis solicitado. Para cada solicitud de un elemento de sintaxis, el decodificador usa el mismo esquema de binarización que se usó para codificar el elemento de sintaxis correspondiente. El esquema de binarización puede ser diferente para diferentes categorías de elementos de sintaxis. El esquema de binarización para una categoría de elementos de sintaxis particular puede depender del conjunto de valores de elementos de sintaxis posibles y/u otras propiedades de los elementos de sintaxis para la categoría particular.

El asignador 316 de parámetros asigna un conjunto de uno o más parámetros a cada solicitud de un elemento binario y envía la solicitud de un elemento binario con el conjunto asociado de parámetros al selector de memoria intermedia de elementos binarios. El conjunto de parámetros que se asignan a un elemento binario solicitado por el asignador de parámetros es el mismo que se asignó al elemento binario correspondiente durante la codificación. El conjunto de parámetros puede consistir en uno o más de los parámetros que se mencionan en la descripción del codificador de la figura 7.

En una realización preferida de la invención, el asignador 316 de parámetros asocia cada solicitud de un elemento binario con los mismos parámetros con los que la asoció el asignador 304, es decir un contexto y su medida asociada de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario solicitado actual, tal como una medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario solicitado actual y un identificador que especifica una estimación de cuál de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable o más probable para el elemento binario solicitado actual.

El asignador 316 de parámetros puede determinar una o más de las medidas de probabilidad anteriormente mencionadas (medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario solicitado actual, medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario solicitado actual, identificador que especifica una estimación de cuál de los dos valores de elemento binario posibles representa el valor de elemento

binario menos probable o más probable para el elemento binario solicitado actual) basándose en un conjunto de uno o más símbolos ya decodificados. La determinación de las medidas de probabilidad para una solicitud particular de un elemento binario copia el proceso en el codificador para el elemento binario correspondiente. Los símbolos decodificados que se usan para determinar las medidas de probabilidad pueden incluir uno o más símbolos ya decodificados de la misma categoría de símbolo, uno o más símbolos ya decodificados de la misma categoría de símbolo que corresponden a conjuntos de datos (tales como bloques o grupos de muestras) de ubicaciones espaciales y/o temporales vecinas (con respecto al conjunto de datos asociado con la solicitud actual de un elemento de sintaxis), o uno o más símbolos ya decodificados de diferentes categorías de símbolo que corresponden a conjuntos de datos de ubicaciones espaciales y/o temporales iguales y/o vecinas (con respecto al conjunto de datos asociado con la solicitud actual de un elemento de sintaxis).

Cada solicitud de un elemento binario con un conjunto asociado de parámetros 317 que se emite del asignador 316 de parámetros se alimenta a un selector 318 de memoria intermedia de elementos binarios. Basándose en el conjunto de parámetros 317 asociado, el selector 318 de memoria intermedia de elementos binarios envía una solicitud de un elemento 319 binario a una de dos o más memorias 320 intermedias de elementos binarios y recibe un elemento 325 binario decodificado de la memoria 320 intermedia de elementos binarios seleccionada. Posiblemente se modifica el elemento 325 binario de entrada decodificado y el elemento 326 binario de salida decodificado (con un valor posiblemente modificado) se envía al binarizador 314 como respuesta final a la solicitud de un elemento binario con un conjunto de parámetros 317 asociado.

La memoria 320 intermedia de elementos binarios a la que se reenvía la solicitud de un elemento binario se selecciona de la misma manera que la memoria intermedia de elementos binarios a la que se envió el elemento binario de salida del selector de memoria intermedia de elementos binarios en el lado de codificador.

En una realización preferida de la invención, el selector 318 de memoria intermedia de elementos binarios determina la memoria 320 intermedia de elementos binarios a la que se envía la solicitud de un elemento 319 binario basándose en la medida asociada de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario solicitado actual. En una realización preferida de la invención, el conjunto de valores posibles para la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles es finito y el selector 318 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 320 intermedia de elementos binarios con cada valor posible de la estimación de la probabilidad para uno de los dos valores de elemento binario posibles, en el que diferentes valores para la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles pueden asociarse con la misma memoria 320 intermedia de elementos binarios. En una realización preferida adicional de la invención, el intervalo de valores posibles para la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles se reparte en varios intervalos, el selector 318 de memoria intermedia de elementos binarios determina el índice de intervalo para la medida actual de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles, y el selector 318 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 320 intermedia de elementos binarios con cada valor posible para el índice de intervalo, en el que diferentes valores para el índice de intervalo pueden asociarse con la misma memoria 320 intermedia de elementos binarios. En una realización preferida de la invención, solicitudes de elementos binarios 317 con medidas opuestas de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles (medidas opuestas son aquellas que representan estimaciones de probabilidad P y 1 - P) se reenvían a la misma memoria 320 intermedia de elementos binarios. En una realización preferida adicional de la invención, la asociación de la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para la solicitud de elemento binario actual con una particular memoria intermedia de elementos binarios se adapta a lo largo del tiempo.

En una realización preferida adicional de la invención, el selector 318 de memoria intermedia de elementos binarios determina la memoria 320 intermedia de elementos binarios a la que se envía la solicitud de un elemento 319 binario basándose en la medida asociada de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario solicitado actual. En una realización preferida de la invención, el conjunto de valores posibles para la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable es finito y el selector 318 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 320 intermedia de elementos binarios con cada valor posible de la estimación de la probabilidad para el valor de elemento binario menos probable o más probable, en el que diferentes valores para la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable pueden asociarse con la misma memoria 320 intermedia de elementos binarios. En una realización preferida adicional de la invención, el intervalo de valores posibles para la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable se reparte en varios intervalos, el selector 318 de memoria intermedia de elementos binarios determina el índice de intervalo para la medida actual de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable, y el selector 318 de memoria intermedia de elementos binarios contiene una tabla que asocia exactamente una memoria 320 intermedia de elementos binarios con cada valor posible para el índice de intervalo, en el que diferentes valores para

el índice de intervalo pueden asociarse con la misma memoria 320 intermedia de elementos binarios. En una realización preferida adicional de la invención, la asociación de la medida de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para la solicitud de elemento binario actual con una memoria intermedia de elementos binarios particular se adapta a lo largo del tiempo.

Tras recibir un elemento 325 binario decodificado desde la he memoria 320 intermedia de elementos binarios seleccionada, el selector 318 de memoria intermedia de elementos binarios modifica posiblemente el elemento 325 binario de entrada y envía el elemento 326 binario de salida (con un valor posiblemente modificado) al binarizador 314. El mapeo de elemento binario de entrada/salida del selector 318 de memoria intermedia de elementos binarios es la inversa del mapeo de elemento binario de entrada/salida del selector de memoria intermedia de elementos binarios en el lado de codificador.

En una realización preferida de la invención, el selector 318 de memoria intermedia de elementos binarios no modifica el valor del elemento binario, es decir, el elemento 326 binario de salida siempre tiene el mismo valor que el elemento 325 binario de entrada. En una realización preferida adicional de la invención, el selector 318 de memoria intermedia de elementos binarios determina el valor 326 de elemento binario de salida basándose en el valor 325 de elemento binario de entrada y la medida de una estimación de la probabilidad para uno de los dos valores de elemento binario posibles para el elemento binario solicitado actual que está asociado con la solicitud de un elemento 317 binario. En una realización preferida de la invención, el valor 326 de elemento binario de salida se establece igual al valor 325 de elemento binario de entrada si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para la solicitud de elemento binario actual es menor que (o menor que o igual a) un umbral particular; si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para la solicitud de elemento binario actual es mayor que o igual a (o mayor que) un umbral particular, el valor 326 de elemento binario de salida se modifica (es decir, se establece al opuesto del valor de elemento binario de entrada). En una realización preferida adicional de la invención, el valor 326 de elemento binario de salida se establece igual al valor 325 de elemento binario de entrada si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para la solicitud de elemento binario actual es mayor que (o mayor que o igual a) un umbral particular; si la medida de la probabilidad para uno de los dos valores de elemento binario posibles para la solicitud de elemento binario actual es menor que o igual a (o menor que) un umbral particular, el valor 326 de elemento binario de salida se modifica (es decir, se establece al opuesto del valor de elemento binario de entrada). En una realización preferida de la invención, el valor del umbral corresponde a un valor de 0,5 para la probabilidad estimada para ambos valores de elemento binario posibles.

En una realización preferida adicional de la invención, el selector 318 de memoria intermedia de elementos binarios determina el valor 326 de elemento binario de salida basándose en el valor 325 de elemento binario de entrada y el identificador, que especifica una estimación de cuál de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable o más probable para la solicitud de elemento binario actual, que se asocia con la solicitud de un elemento 317 binario. En una realización preferida de la invención, el valor 326 de elemento binario de salida se establece igual al valor 325 de elemento binario de entrada si el identificador especifica que el primero de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable (o más probable) para la solicitud de elemento binario actual, y el valor 326 de elemento binario de salida se modifica (es decir, se establece al opuesto del valor de elemento binario de entrada) si el identificador especifica que el segundo de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable (o más probable) para la solicitud de elemento binario actual.

Tal como se describió anteriormente, el selector de memoria intermedia de elementos binarios envía una solicitud de un elemento 319 binario a una de las dos o más memorias 320 intermedias de elementos binarios. Las memorias 20 intermedias de elementos binarios representan memorias intermedias de primero en entrar primero en salir, que se alimentan con secuencias de elementos 321 binarios decodificados desde los decodificadores 322 de elementos binarios conectados. Como respuesta a una solicitud de un elemento 319 binario que se envía a una memoria 320 intermedia de elementos binarios desde el selector 318 de memoria intermedia de elementos binarios, la memoria 320 intermedia de elementos binarios retira de su contenido el elemento binario que se alimentó en primer lugar en la memoria 320 intermedia de elementos binarios y lo envía al selector 318 de memoria intermedia de elementos binarios. Los elementos binarios que se envían más temprano a la memoria 320 intermedia de elementos binarios se retiran más temprano y se envían al selector 318 de memoria intermedia de elementos binarios.

Cada una de las dos o más memorias 320 intermedias de elementos binarios está conectada con exactamente un decodificador 322 de elementos binarios y cada decodificador de elementos binarios solo está conectado con una memoria 320 intermedia de elementos binarios. Cada decodificador 322 de elementos binarios lee palabras 323 de código, que representan secuencias de bits, a partir de un flujo 324 de bits parcial independiente. El decodificador de elementos binarios convierte una palabra 323 de código en una secuencia de elementos 321 binarios que se envía a la memoria 320 intermedia de elementos binarios conectada. El algoritmo de decodificación global convierte dos o más flujos 324 de bits parciales en varios elementos de sintaxis decodificados, en el que el número de flujos de bits parciales es igual al número de memorias intermedias de elementos binarios y decodificadores de elementos

binarios y la decodificación de elementos de sintaxis se activa mediante solicitudes de nuevos elementos de sintaxis. En una realización preferida de la invención, un decodificador 322 de elementos binarios convierte palabras 323 de código de un número variable de bits en una secuencia de un número variable de elementos 321 binarios. Una ventaja de realizaciones de la invención es que la decodificación de elementos binarios a partir de los dos o más flujos de bits parciales puede realizarse en paralelo (por ejemplo, para diferentes grupos de medidas de probabilidad), lo cual reduce el tiempo de procesamiento para varias implementaciones.

Otra ventaja de realizaciones de la invención es que la decodificación de elementos binarios, que se realiza mediante los decodificadores 322 de elementos binarios, puede diseñarse específicamente para diferentes conjuntos de parámetros 317. En particular, la codificación y decodificación de elementos binarios pueden optimizarse (en cuanto a eficiencia de codificación y/o complejidad) para diferentes grupos de probabilidades estimadas. Por un lado, esto permite una reducción de la complejidad de codificación/decodificación con respecto a algoritmos de codificación entrópica del estado de la técnica con eficiencia de codificación similar. Por otro lado, permite una mejora de la eficiencia de codificación con respecto a algoritmos de codificación entrópica del estado de la técnica con complejidad de codificación/decodificación similar. En una realización preferida de la invención, los decodificadores 322 de elementos binarios implementan diferentes algoritmos de decodificación (es decir mapeo de secuencias de elementos binarios en palabras de código) para diferentes grupos de medidas de una estimación de la probabilidad para uno de los dos valores 317 de elemento binario posibles para la solicitud de elemento binario actual. En una realización preferida adicional de la invención, los decodificadores 322 de elementos binarios implementan diferentes algoritmos de decodificación para diferentes grupos de medidas de una estimación de la probabilidad para el valor de elemento binario menos probable o más probable para el elemento binario solicitado actual.

Los decodificadores 322 de elementos binarios realizan el mapeo inverso a los codificadores de elementos binarios correspondientes en el lado de codificador.

En una realización preferida de la invención, los decodificadores 322 de elementos binarios (o uno o más de los decodificadores de elementos binarios) representan decodificadores entrópicos que mapean directamente palabras 323 de código en secuencias de elementos 321 binarios. Tales mapeos pueden implementarse de manera eficiente y no requieren un motor de codificación aritmética complejo. El mapeo de palabras de código en secuencias de elementos binarios tiene que ser único. En una realización preferida de la invención, el mapeo de palabras 323 de código en secuencias de elementos 321 binarios es biyectivo. En una realización preferida adicional de la invención, los decodificadores 310 de elementos binarios (o uno o más de los decodificadores de elementos binarios) representan decodificadores entrópicos que mapean directamente palabras 323 de código de longitud variable en secuencias de longitud variable de elementos 321 binarios. En una realización preferida de la invención, las palabras de código de entrada representan códigos libres de redundancia tales como códigos de Huffman generales o códigos de Huffman canónicos. En la tabla 3 se ilustran dos ejemplos del mapeo biyectivo de códigos libres de redundancia en secuencias de elementos binarios.

En una realización preferida adicional de la invención, los decodificadores 322 de elementos binarios (o uno o más de los decodificadores de elementos binarios) representan decodificadores entrópicos que mapean directamente palabras 323 de código de longitud fija en secuencias de longitud variable de elementos 321 binarios. En una realización preferida adicional de la invención, los decodificadores 322 de elementos binarios (o uno o más de los decodificadores de elementos binarios) representan decodificadores entrópicos que mapean directamente palabras 323 de código de longitud variable en secuencias de longitud fija de elementos 321 binarios.

Por tanto, las figuras 7 y 8 mostraban una realización de un codificador para codificar una secuencia de símbolos 3 y un decodificador para reconstruir la misma. El codificador comprende un asignador 304 configurado para asignar varios parámetros 305 a cada símbolo de la secuencia de símbolos. La asignación se basa en información contenida dentro de símbolos anteriores de la secuencia de símbolos tal como la categoría del elemento 1 de sintaxis a cuya representación (tal como binarización) pertenece el símbolo actual y que, según la estructura de sintaxis de los elementos 1 de sintaxis, se espera actualmente, expectativa que, a su vez, puede deducirse a partir del historial de elementos 1 de sintaxis y símbolos 3 anteriores. Además, el codificador comprende una pluralidad de codificadores 10 entrópicos cada uno de los cuales está configurado para convertir los símbolos 3 reenviados al codificador entrópico respectivo en un flujo 312 de bits respectivo, y un selector 306 configurado para reenviar cada símbolo 3 a uno seleccionado de la pluralidad de codificadores 10 entrópicos, dependiendo la selección del número de parámetros 305 asignados al símbolo 3 respectivo. Puede considerarse que el asignador 304 está integrado en el selector 206 con el fin de proporcionar un selector 502 respectivo.

El decodificador para reconstruir una secuencia de símbolos comprende una pluralidad de decodificadores 322 entrópicos, cada uno de los cuales está configurado para convertir un flujo 323 de bits respectivo en símbolos 321; un asignador 316 configurado para asignar varios parámetros 317 a cada símbolo 315 de una secuencia de símbolos que va a reconstruirse basándose en información contenida dentro de símbolos anteriormente reconstruidos de la secuencia de símbolos (véanse 326 y 327 en la figura 8); y un selector 318 configurado para

recuperar cada símbolo de la secuencia de símbolos que va a reconstruirse a partir de uno seleccionado de la pluralidad de decodificadores 322 entrópicos, dependiendo la selección del número de parámetros definidos en el símbolo respectivo. El asignador 316 puede estar configurado de tal manera que el número de parámetros asignados a cada símbolo comprende, o es, una medida de una estimación de una probabilidad de distribución entre los valores de símbolo posibles que puede adoptar un símbolo respectivo. De nuevo, puede considerarse que el asignador 316 y el selector 318 están integrados en un bloque, un selector 402. La secuencia de símbolos que va a reconstruirse puede ser de un alfabeto binario y el asignador 316 puede estar configurado de tal manera que la estimación de la probabilidad distribución consiste en una medida de una estimación de una probabilidad de un valor de elemento binario menos probable o más probable de los dos valores de elemento binario posibles del alfabeto binario y un identificador que especifica una estimación de cuál de los dos valores de elemento binario posibles representa el valor de elemento binario menos probable o más probable. El asignador 316 puede estar configurado además para asignar de manera interna un contexto a cada símbolo de la secuencia de símbolos 315 que va a reconstruirse basándose en la información contenida dentro de símbolos anteriormente reconstruidos de la secuencia de símbolos que va a reconstruirse teniendo cada contexto una estimación de distribución de probabilidad respectiva asociada con el mismo, y para adaptar la estimación de distribución de probabilidad para cada contexto a una estadística de símbolo real basándose en valores de símbolo de símbolos anteriormente reconstruidos a los que se asigna el contexto respectivo. El contexto puede tener en cuenta una relación espacial o de vecindario de posiciones al que pertenecen los elementos de sintaxis tal como en codificación de vídeo o imágenes, o incluso en tablas en el caso de aplicaciones financieras. Entonces, la medida de la estimación de la distribución de probabilidad para cada símbolo puede determinarse basándose en la estimación de distribución de probabilidad asociada con el contexto asignado al símbolo respectivo tal como mediante cuantificación, o usando como índice en una tabla respectiva, la estimación de distribución de probabilidad asociada con el contexto asignado con el símbolo respectivo (en las siguientes realizaciones indexado mediante un índice de PIPE junto con un índice de refinamiento) a uno de una pluralidad de representantes de estimación de distribución de probabilidad (eliminando mediante el recorte del índice de refinamiento) con el fin de obtener la medida de la estimación de la distribución de probabilidad (indexando el índice de PIPE el flujo 312 de bits parcial). El selector puede estar configurado de tal manera que se define una asociación biyectiva entre la pluralidad de codificadores entrópicos y la pluralidad de representantes de estimación de distribución de probabilidad. El selector 18 puede estar configurado para cambiar un mapeo de cuantificación desde un intervalo de las estimaciones de distribución de probabilidad hasta la pluralidad de representaciones de estimación de distribución de probabilidad de una manera determinista predeterminada dependiendo de símbolos anteriormente reconstruidos de la secuencia de símbolos, a lo largo del tiempo. Es decir, el selector 318 puede cambiar los tamaños de etapa de cuantificación, es decir los intervalos de distribuciones de probabilidad mapeados en los índices de probabilidad individuales asociados de manera biyectiva con los decodificadores entrópicos individuales. La pluralidad de decodificadores 322 entrópicos, a su vez, pueden estar configurados para adaptar su manera de convertir símbolos en flujos de bits sensibles a un cambio en el mapeo de cuantificación. Por ejemplo, cada decodificador 322 entrópico puede estar optimizado, es decir puede tener una tasa de compresión óptima, para una determinada estimación de distribución de probabilidad dentro del intervalo de cuantificación de estimación de distribución de probabilidad respectiva, y puede cambiar su mapeo de palabra de código/secuencia de símbolos para adaptar la posición de esta determinada estimación de distribución de probabilidad dentro del intervalo de cuantificación de estimación de distribución de probabilidad respectiva tras un cambio de este último para optimizarse. El selector puede estar configurado para cambiar el mapeo de cuantificación de tal manera que tasas mediante las cuales se recuperan los símbolos de la pluralidad de decodificadores entrópicos se vuelven menos dispersas. En cuanto al binarizador 314, se observa que puede omitirse si los elementos de sintaxis ya son binarios. Además, dependiendo del tipo de decodificador 322, la existencia de las memorias 320 intermedias no es necesaria. Además, las memorias intermedias pueden estar integradas dentro de los decodificadores.

Terminación de secuencias de elementos de sintaxis finitas

En una realización preferida de la invención, la codificación y decodificación se realizan para un conjunto finito de elementos de sintaxis. Con frecuencia se codifica una determinada cantidad de datos tales como una imagen fija, una trama o campo de una secuencia de vídeo, un segmento de una imagen, un segmento de una trama o un campo de una secuencia de vídeo, o un conjunto de muestras de audio sucesivas, etc. Para conjuntos finitos de elementos de sintaxis, en general, tienen que terminarse los flujos de bits parciales que se crean en el lado de codificador, es decir, tiene que garantizarse que todos los elementos de sintaxis pueden decodificarse a partir de los flujos de bits parciales transmitidos o almacenados. Tras insertarse el último elemento binario en la memoria 308 intermedia de elementos binarios correspondiente, el codificador 310 de elementos binarios tiene que garantizar que se escribe una palabra de código completa en el flujo 312 de bits parcial. Si el codificador 310 de elementos binarios representa un codificador entrópico que implementa un mapeo directo de secuencias de elementos binarios en palabras de código, la secuencia de elementos binarios que se almacena en la memoria intermedia de elementos binarios tras escribir el último elemento binario en la memoria intermedia de elementos binarios puede no representar una secuencia de elementos binarios que está asociada con una palabra de código (es decir, puede representar un prefijo de dos o más secuencias de elementos binarios que están asociadas con palabras de código). En tal caso, cualquiera de las palabras de código asociadas con una secuencia de elementos binarios que contiene

la secuencia de elementos binarios en la memoria intermedia de elementos binarios como prefijo tiene que escribirse en el flujo de bits parcial (la memoria intermedia de elementos binarios tiene que alinearse). Esto puede realizarse insertando elementos binarios con un valor particular o arbitrario en la memoria intermedia de elementos binarios hasta que se escribe una palabra de código. En una realización preferida de la invención, el codificador de elementos binarios selecciona una de las palabras de código con longitud mínima (además de la propiedad de que la secuencia de elementos binarios asociada debe contener la secuencia de elementos binarios en la memoria intermedia de elementos binarios como prefijo). En el lado de decodificador, el decodificador 322 de elementos binarios puede decodificar más elementos binarios que los requeridos para la última palabra de código en un flujo de bits parcial; estos elementos binarios no los solicita el selector 318 de memoria intermedia de elementos binarios y se desechan e ignoran. La decodificación del conjunto finito de símbolos se controla mediante solicitudes de elementos de sintaxis decodificados; si no se solicita ningún elemento de sintaxis adicional para una cantidad de datos, se termina la decodificación.

Transmisión y multiplexado de los flujos de bits parciales

Los flujos 312 de bits parciales que se crean mediante el codificador pueden transmitirse por separado, o pueden multiplexarse para dar un único flujo de bits, o las palabras de código de los flujos de bits parciales pueden entrelazarse para dar un único flujo de bits.

En una realización de la invención, cada flujo de bits parcial para una cantidad de datos se escribe en un paquete de datos. La cantidad de datos puede ser un conjunto arbitrario de elementos de sintaxis tales como una imagen fija, un campo o trama de una secuencia de vídeo, un segmento de una imagen fija, un segmento de un campo o trama de una secuencia de vídeo, o una trama de muestras de audio, etc.

En otra realización preferida de la invención, dos o más de los flujos de bits parciales para una cantidad de datos o todos los flujos de bits parciales para una cantidad de datos se multiplexan para dar un paquete de datos. En la figura 9 se ilustra la estructura de un paquete de datos que contiene flujos de bits parciales multiplexados.

El paquete 400 de datos consiste en una cabecera y una partición para los datos de cada flujo de bits parcial (para la cantidad de datos considerada). La cabecera 400 del paquete de datos contiene indicaciones para el reparto del (resto del) paquete de datos en segmentos de datos 402 de flujo de bits. Además de las indicaciones para el reparto, la cabecera puede contener información adicional. En una realización preferida de la invención, las indicaciones para el reparto del paquete de datos son las ubicaciones del comienzo de los segmentos de datos en unidades de bits o bytes o múltiplos de bits o múltiplos de bytes. En una realización preferida de la invención, las ubicaciones del comienzo de los segmentos de datos se codifican como valores absolutos en la cabecera del paquete de datos, o bien con respecto al comienzo del paquete de datos o bien con respecto al final de la cabecera o bien con respecto al comienzo del paquete de datos anterior. En una realización preferida adicional de la invención, las ubicaciones del comienzo de los segmentos de datos se codifican de manera diferencial, es decir, solo se codifica la diferencia entre el comienzo real de un segmento de datos y una predicción del comienzo de los segmento de datos. La predicción puede derivarse basándose en información ya conocida o transmitida tal como el tamaño global del paquete de datos, el tamaño de la cabecera, el número de segmentos de datos en el paquete de datos, la ubicación del comienzo de segmentos de datos anteriores. En una realización preferida de la invención, la ubicación del comienzo del primer paquete de datos no se codifica, sino que se deduce basándose en el tamaño de la cabecera de paquete de datos. En el lado de decodificador, las indicaciones de partición transmitidas se usan para derivar el comienzo de los segmentos de datos. Después se usan los segmentos de datos como flujos de bits parciales y los datos contenidos en los segmentos de datos se alimentan a los decodificadores de elementos binarios correspondientes en orden secuencial.

Hay varias alternativas para multiplexar los flujos de bits parciales para dar un paquete de datos. Una alternativa, que puede reducir la información secundaria requerida, en particular para casos en los que los tamaños de los flujos de bits parciales son muy similares, se ilustra en la figura 10. La carga útil del paquete de datos, es decir, el paquete 410 de datos sin su cabecera 411, se reparte en segmentos 412 de una manera predefinida. Como ejemplo, la carga útil de paquete de datos puede repartirse en segmentos del mismo tamaño. Después se asocia cada segmento con un flujo de bits parcial o con la primera parte de un flujo 413 de bits parcial. Si un flujo de bits parcial es mayor que el segmento de datos asociado, su resto 414 se coloca en el espacio sin usar al final de otros segmentos de datos. Esto puede realizarse de una manera tal que la parte restante de un flujo de bits se inserta en orden inverso (comenzando desde el final de los segmentos de datos), lo cual reduce la información secundaria. La asociación de los restos de los flujos de bits parciales con segmentos de datos y, cuando se añade más de un resto a un segmento de datos, el punto de inicio para uno o más de los restos, tienen que indicarse dentro del flujo de bits, por ejemplo, en la cabecera de paquete de datos.

Entrelazado de palabras de código de longitud variable

Para algunas aplicaciones, el multiplexado anteriormente descrito de los flujos de bits parciales (para una cantidad

de elementos de sintaxis) en un paquete de datos puede tener las siguientes desventajas: por un lado, para paquetes de datos pequeños, el número de bits para la información secundaria que se requiere para indicar el reparto puede volverse significativo con respecto a los datos reales en los flujos de bits parciales, lo cual reduce finalmente la eficiencia de codificación. Por otro lado, el multiplexado puede no ser adecuado para aplicaciones que requieren un bajo retardo (por ejemplo, para aplicaciones de videoconferencia). Con el multiplexado descrito, el codificador no puede comenzar la transmisión de un paquete de datos antes de que se hayan creado completamente los flujos de bits parciales, dado que las ubicaciones del comienzo de las particiones no se conocen previamente. Además, en general, el decodificador tiene que esperar hasta que recibe el comienzo del último segmento de datos antes de que pueda empezar la decodificación de un paquete de datos. Para aplicaciones tales como sistemas de videoconferencia, estos retardos pueden añadirse para dar un retardo global adicional del sistema de varias imágenes de vídeo (en particular para tasas de transferencia de bits que son próximas a la tasa de transferencia de bits de transmisión y para codificadores/decodificadores que requieren casi el intervalo de tiempo entre dos imágenes para codificar/decodificar una imagen), lo cual resulta crítico para tales aplicaciones. Con el fin de superar las desventajas para determinadas aplicaciones, el codificador de una realización preferida de la invención puede estar configurado de una manera tal que las palabras de código que se generan mediante los dos o más codificadores de elementos binarios se entrelazan para dar un único flujo de bits. El flujo de bits con las palabras de código entrelazadas puede enviarse directamente al decodificador (cuando se desprecia un pequeño retardo de memoria intermedia, véase a continuación). En el lado de decodificador, los dos o más decodificadores de elementos binarios leen las palabras de código directamente a partir del flujo de bits en orden de decodificación; la decodificación puede iniciarse con el primer bit recibido. Además, no se requiere nada de información secundaria para indicar el multiplexado (o entrelazado) de los flujos de bits parciales. Una manera adicional de reducir la complejidad de decodificador puede lograrse cuando los decodificadores 322 de elementos binarios no leen palabras de código de longitud variable a partir de una memoria intermedia de bits global, sino que en vez de eso siempre leen secuencias de longitud fija de bits a partir de la memoria intermedia de bits global y añaden estas secuencias de longitud fija de bits a una memoria intermedia de bits local, en la que cada decodificador 322 de elementos binarios está conectado con una memoria intermedia de bits local independiente. Después se leen las palabras de código de longitud variable a partir de la memoria intermedia de bits local. Por tanto, el análisis sintáctico de palabras de código de longitud variable puede realizarse en paralelo, solo el acceso de secuencias de longitud fija de bits tiene que realizarse de una manera sincronizada, pero tal acceso de secuencias de longitud fija de bits es habitualmente muy rápido, de modo que puede reducirse la complejidad de decodificación global para algunas arquitecturas. El número fijo de elementos binarios que se envían a una memoria intermedia de bits local particular puede ser diferente para diferentes memorias intermedias de bits locales y también puede variar a lo largo del tiempo, dependiendo de determinados parámetros tales como acontecimientos en el decodificador de elementos binarios, memoria intermedia de elementos binarios, o memoria intermedia de bits. Sin embargo, el número de bits que se leen mediante un acceso particular no depende de los bits reales que se leen durante el acceso particular, lo cual es la diferencia importante con respecto a la lectura de palabras de código de longitud variable. La lectura de las secuencias de longitud fija de bits se activa mediante determinados acontecimientos en las memorias intermedias de elementos binarios, decodificadores de elementos binarios o memorias intermedias de bits locales. Como ejemplo, es posible solicitar la lectura de una nueva secuencia de longitud fija de bits cuando el número de bits que están presentes en una memoria intermedia de bits conectada disminuye por debajo de un umbral predefinido, en el que pueden usarse diferentes valores de umbral para diferentes memorias intermedias de bits. En el codificador, tiene que garantizarse que las secuencias de longitud fija de elementos binarios se insertan en el mismo orden en el flujo de bits en el que se leen a partir del flujo de bits en el lado de decodificador. También es posible combinar este entrelazado de secuencias de longitud fija con un control de bajo retardo similar a los explicados anteriormente. A continuación, se describe una realización preferida para el entrelazado de secuencias de longitud fija de bits. Para detalles adicionales con respecto a los últimos esquemas de entrelazado, se hace referencia al documento WO2011/128268A1.

Tras haberse descrito realizaciones según las cuales se usa la codificación incluso anterior para comprimir datos de vídeo, se describe una realización incluso adicional para implementar realizaciones de la presente invención que hacen que la implementación sea especialmente eficaz en cuanto a un buen compromiso entre tasa de compresión por un lado y tabla de consulta y coste de cálculo por otro lado. En particular, las siguientes realizaciones permiten el uso de códigos de longitud variable computacionalmente menos complejos con el fin de codificar de manera entrópica los flujos de bits individuales, y cubrir eficazmente porciones de la estimación de probabilidad. En las realizaciones descritas a continuación, los símbolos son de naturaleza binaria y los códigos de VLC presentados a continuación cubren eficazmente la estimación de probabilidad representada, por ejemplo, mediante R_{LPS}, que se extiende dentro de [0;0,5].

En particular, las realizaciones explicadas resumidamente a continuación describen posibles implementaciones para los codificadores 310 y decodificadores 322 entrópicos individuales en las figuras 7 a 17, respectivamente. Son adecuadas para la codificación de elementos binarios, es decir símbolos binarios, tal como se producen en aplicaciones de compresión de imágenes o vídeos. Por consiguiente, estas realizaciones también son aplicables a la codificación de imágenes o vídeos en la que tales símbolos binarios se dividen en el uno o más flujos de elementos 307 binarios que van a codificarse y flujos 324 de bits que van a decodificarse, respectivamente, en las que cada

uno de tales flujos de elementos binarios puede considerarse como una realización de un proceso de Bernoulli. Las realizaciones descritas a continuación usan uno o más de los diversos códigos denominados variable a variable (códigos v2v) explicados a continuación para codificar los flujos de elementos binarios. Un código v2v puede considerarse como dos códigos libres de prefijo con el mismo número de palabras de código. Un código libre de prefijo primario y uno secundario. Cada palabra de código del código libre de prefijo primario se asocia con una palabra de código del código libre de prefijo secundario. Según las realizaciones explicadas resumidamente a continuación, al menos algunos de los codificadores 310 y decodificadores 322 funcionan de la siguiente manera: para codificar una secuencia particular de elementos 307 binarios, siempre que se lee una palabra de código del código libre de prefijo primario a partir de la memoria 308 intermedia, la palabra de código correspondiente del código libre de prefijo secundario se escribe en el flujo 312 de bits. Se usa el mismo procedimiento para decodificar un flujo 324 de bits de este tipo, pero intercambiándose los códigos libres de prefijo primario y secundario. Es decir, para decodificar un flujo 324 de bits, siempre que se lee una palabra de código del código libre de prefijo secundario a partir del flujo 324 de bits respectivo, se escribe la palabra de código correspondiente del código libre de prefijo primario en la memoria 320 intermedia.

Ventajosamente, los códigos descritos a continuación no necesitan tablas de consulta. Los códigos pueden implementarse en forma de máquinas de estados finitos. Los códigos v2v presentados en este caso pueden generarse mediante reglas de construcción sencillas de tal manera que no hay necesidad de almacenar grandes tablas para las palabras de código. En lugar de eso, puede usarse un algoritmo sencillo para llevar a cabo la codificación o decodificación. A continuación, se describen tres reglas de construcción en las que dos de ellas pueden parametrizarse. Cubren porciones diferentes o incluso independientes del intervalo de probabilidad anteriormente mencionado y, por consiguiente, son específicamente ventajosas si se usan en conjunto, tal como los tres códigos en paralelo (cada uno para unos diferentes de los codificadores/decodificadores 11 y 22), o dos de ellos. Con las reglas de construcción descritas a continuación, es posible diseñar un conjunto de códigos v2v, de tal manera que para procesos de Bernoulli con probabilidad arbitraria p, uno de los códigos funciona bien en cuanto a longitud de código en exceso.

Tal como se mencionó anteriormente, la codificación y decodificación de los flujos 312 y 324 respectivamente pueden realizarse o bien de manera independiente para cada flujo o bien de una manera entrelazada. Sin embargo, esto no es específico de las clases presentadas de códigos v2v, y por tanto a continuación solo se describe la codificación y decodificación de una palabra de código particular para cada una de las tres reglas de construcción. Sin embargo, se enfatiza que todas las realizaciones anteriores referentes a las soluciones de entrelazado también pueden combinarse con los códigos actualmente descritos o los codificadores y decodificadores 310 y 322, respectivamente.

Regla de construcción 1: Códigos de “PIPE de elementos binarios unarios” o codificadores/decodificadores 310 y 322

Los códigos de PIPE de elementos binarios unarios (PIPE = entropía de reparto de intervalo de probabilidad) son una versión especial de los denominados códigos “PIPE de elementos binarios”, es decir códigos adecuados para la codificación de cualquiera de los flujos 12 y 24 de bits individuales, que transfieren cada uno datos de unas estadísticas de símbolo binarias pertenecientes a un determinado subintervalo de probabilidad del intervalo de probabilidad anteriormente mencionado [0;0,5]. En primer lugar, se describe la construcción de códigos de PIPE de elementos binarios. Puede construirse un código de PIPE de elementos binarios a partir de cualquier código libre de prefijo con al menos tres palabras de código. Para formar un código v2v, usa el código libre de prefijo como código primario y secundario, pero con dos palabras de código del código libre de prefijo secundario intercambiadas. Esto significa que, excepto para dos palabras de código, los elementos binarios se escriben en el flujo de bits sin cambiarse. Con esta técnica, solo se necesita almacenar un código libre de prefijo junto con la información de que dos palabras de código están intercambiadas y, por tanto, se reduce el consumo de memoria. Obsérvese que solo tiene sentido intercambiar palabras de código de diferente longitud ya que, de lo contrario, el flujo de bits tendría la misma longitud que el flujo de elementos binarios (despreciando efectos que pueden producirse al final del flujo de elementos binarios).

Debido a esta regla de construcción, una propiedad destacada uno de los códigos de PIPE de elementos binarios es que, si se intercambian los códigos libres de prefijo primario y secundario (mientras se conserva el mapeo de las palabras de código), el código v2v resultante es idéntico al código v2v original. Por tanto, el algoritmo de codificación y el algoritmo de decodificación son idénticos para códigos de PIPE de elementos binarios.

Se construye un código de PIPE de elementos binarios unario a partir de un código libre de prefijo especial. Este código libre de prefijo especial se construye de la siguiente manera. En primer lugar, se genera un código libre de prefijo que consiste en n palabras de código unario empezando con “01”, “001”, “0001”, ... hasta que se producen n palabras de código. n es el parámetro para el código de PIPE de elementos binarios unario. A partir de la palabra de código más larga, se elimina el último 1. Esto corresponde a un código unario truncado (pero sin la palabra de código “0”). Después, se generan n - 1 palabras de código unario empezando con “10”, “110”, “1110”, ... hasta que

se producen n - 1 palabras de código. A partir de la más larga de estas palabras de código, se elimina el último 0. El conjunto de unión de estos dos códigos libres de prefijo se usa como entrada para generar el código de PIPE de elementos binarios unario. Las dos palabras de código que se intercambian son la que solo consiste en ceros y la que sólo consiste en unos.

Ejemplo para n = 4:

N.º	Primario	Secundario
1	0000	111
2	0001	0001
3	001	001
4	01	01
5	10	10
6	110	110
7	111	0000

Regla de construcción 2: Códigos “de unarios a Rice” y codificadores/decodificadores 10 y 22 de unarios a Rice:

Los códigos de unarios a Rice usan un código unario truncado como código primario. Es decir, se generan palabras de código unario empezando con “1”, “01”, “001”, ... hasta que se generan 2” + 1 palabras de código y a partir de la palabra de código más larga, se elimina el último 1. n es el parámetro del código de unario a Rice. El código libre de prefijo secundario se construye a partir de las palabras de código del código libre de prefijo primario de la siguiente manera. A la palabra de código primaria que solo consiste en ceros se le asigna la palabra de código “1”. Todas las demás palabras de código consisten en la concatenación de la palabra de código “0” con la representación binaria de n bits del número de ceros de la palabra de código correspondiente del código libre de prefijo primario.

Ejemplo para n = 3:

N.º	Primario	Secundario
1	1	0000
2	01	0001
3	001	0010
4	0001	0011
5	00001	0100
6	000001	0101
7	0000001	0110
8	00000001	0111
9	00000000	1
Obsérvese que esto es idéntico a mapear un código unario infinito a un código de Rice con un parámetro de Rice 2^n.		

Regla de construcción 3: Código de “tres elementos binarios”

El código de tres elementos binarios viene dado como:

N.º	Primario	Secundario
1	000	0
2	001	100
3	010	101
4	100	110
5	110	11100
6	101	11101
7	011	11110
8	111	11111

Tiene la propiedad de que el código primario (secuencias de símbolos) tiene longitud fija (siempre tres elementos binarios) y las palabras de código se clasifican mediante números crecientes de unos.

A continuación, se describe una implementación eficiente del código de tres elementos binarios. Un codificador y decodificador para el código de tres elementos binarios pueden implementarse sin almacenar tablas de la siguiente manera.

En el codificador (cualquiera de 10), se leen tres elementos binarios a partir del flujo de elementos binarios (es decir, 7). Si estos tres elementos binarios contienen exactamente un 1, se escribe la palabra de código “1” en el flujo de

bits seguido por dos elementos binarios que consisten en la representación binaria de la posición del 1 (empezando desde la derecha con 00). Si los tres elementos binarios contienen exactamente un 0, se escribe la palabra de código “111” en el flujo de bits seguido por dos elementos binarios que consisten en la representación binaria de la posición del 0 (empezando desde la derecha con 00). Las palabras de código restantes “000” y “111” se mapean a “0” y “11111”, respectivamente.

En el decodificador (cualquiera de 22), se lee elemento binario o bit a partir del flujo 24 de bits respectivo. Si es igual a “0”, se decodifica la palabra de código “000” en el flujo 21 de elementos binarios. Si es igual a “1”, se leen dos elementos binarios más a partir del flujo 24 de bits. Si estos dos bits no son iguales a “11”, se interpretan como la representación binaria de un número y se decodifican dos 0 y un 1 en el flujo de bits de tal manera que la posición del 1 se determina mediante el número. Si los dos bits son iguales a “11”, se leen dos bits más y se interpretan como la representación binaria de un número. Si este número es menor de 3, se decodifican dos 1 y un 0 y el número determina la posición del 0. Si es igual a 3, se decodifica “111” en el flujo de elementos binarios.

A continuación, se describe una implementación eficiente de códigos de PIPE de elementos binarios unarios. Un codificador y decodificador para códigos de PIPE de elementos binarios unarios pueden implementarse de manera eficiente usando un contador. Debido a la estructura de códigos de PIPE de elementos binarios, la codificación y decodificación de códigos de PIPE de elementos binarios son fáciles de implementar:

En el codificador (cualquiera de 10), si el primer elemento binario de una palabra de código es igual a “0”, se procesan elementos binarios hasta que se produce un “1” o hasta que se leen n ceros (incluyendo el primer “0” de la palabra de código). Si se produce un “1”, se escriben los elementos binarios leídos en el flujo de bits sin cambiar. De lo contrario (es decir, se leen n ceros), se escriben n - 1 unos en el flujo de bits. Si el primer elemento binario de la palabra de código es igual a “1”, se procesan elementos binarios hasta que se produce un “0” o hasta que se leen n - 1 unos (incluyendo el primer “1” de la palabra de código). Si se produce un “0”, se escriben los elementos binarios leídos en el flujo de bits sin cambiar. De lo contrario (es decir, se leen n - 1 unos), se escriben n ceros en el flujo de bits.

En el decodificador (cualquiera de 322), se usa el mismo algoritmo que para el codificador, dado que es el mismo para códigos de PIPE de elementos binarios tal como se describió anteriormente.

A continuación, se describe una implementación eficiente de códigos de unarios a Rice. Un codificador y decodificador para códigos de unarios a Rice pueden implementarse de manera eficiente usando un contador tal como se describirá ahora.

En el codificador (cualquiera de 310), se leen elementos binarios a partir del flujo de elementos binarios (es decir 7) hasta que se produce un 1 o hasta que se leen 2^n ceros. Se cuenta el número de ceros. Si el número contado es igual a 2^n, se escribe la palabra de código “1” en el flujo de bits. De lo contrario, se escribe “0”, seguido por la representación binaria del número contado, escrito con n bits.

En el decodificador (cualquiera de 322), se lee un bit. Si es igual a “1”, se decodifican 2^n ceros en la cadena de elementos binarios. Si es igual a “0”, se leen n bits más y se interpretan como representación binaria de un número. Este número de ceros se decodifica en el flujo de elementos binarios, seguido por un “1”.

Dicho de otro modo, las realizaciones que acaban de describirse describen un codificador para codificar una secuencia de símbolos 303, que comprende un asignador 316 configurado para asignar varios parámetros 305 a cada símbolo de la secuencia de símbolos basándose en información contenida dentro de símbolos anteriores de la secuencia de símbolos; una pluralidad de codificadores 310 entrópicos cada uno de los cuales está configurado para convertir los símbolos 307 reenviados al codificador 310 entrópico respectivo en un flujo 312 de bits respectivo; y un selector 6 configurado para reenviar cada símbolo 303 a uno seleccionado de la pluralidad de codificadores 10 entrópicos, dependiendo la selección del número de parámetros 305 asignados al símbolo 303 respectivo. Según las realizaciones que acaban de explicarse resumidamente, al menos un primer subconjunto de los codificadores entrópicos pueden ser un codificador de longitud variable configurado para mapear secuencias de símbolos de longitudes variables dentro del flujo de símbolos 307 en palabras de código de longitudes variables que van a insertarse en el flujo 312 de bits, respectivamente, usando cada uno de los codificadores 310 entrópicos del primer subconjunto una regla de mapeo biyectivo según la cual palabras de código de un código libre de prefijo primario con $(2n-1) \geq 3$ palabras de código se mapean en palabras de código de un código libre de prefijo secundario que es idéntico al código de prefijo primario de tal manera que todas salvo dos de las palabras de código del código libre de prefijo primario se mapean en palabras de código idénticas del código libre de prefijo secundario mientras que las dos palabras de código de los códigos libres de prefijo primario y secundario tienen diferentes longitudes y se mapean entre sí de una manera intercambiada, en la que los codificadores entrópicos pueden usar diferentes n para cubrir diferentes porciones de un intervalo del intervalo de probabilidad anteriormente mencionado. El primer código libre de prefijo puede construirse de tal manera que las palabras de código del primer código libre de prefijo son $(a,b)_2$, $(a,a,b)_3$, ..., $(a,...,a,b)_n$, $(a,...,a)_n$, $(b,a)_2$, $(b,b,a)_3$, ..., $(b,...,b,a)_{n-1}$, $(b,...,b)_{n-1}$, y las dos palabras de código

mapeadas entre sí de la manera intercambiada son $(a,...,a)_n$ y $(b,...,b)_{n-1}$ con $b \neq a$ y $a,b \in \{0,1\}$. Sin embargo, son viables otras alternativas.

Dicho de otro modo, cada uno de un primer subconjunto de codificadores entrópicos puede estar configurado para, al convertir los símbolos reenviados al codificador entrópico respectivo en el flujo de bits respectivo, examinar un primer símbolo reenviado al codificador entrópico respectivo, para determinar si (1) el primer símbolo es igual a $a \in \{0,1\}$, en cuyo caso el codificador entrópico respectivo está configurado para examinar los siguientes símbolos reenviados al codificador entrópico respectivo para determinar si (1.1) se produce b con $b \neq a$ y $b \in \{0,1\}$ dentro de los siguientes n-1 símbolos tras el primer símbolo, en cuyo caso el codificador entrópico respectivo está configurado para escribir una palabra de código en el flujo de bits respectivo, que es igual al primer símbolo seguido por los siguientes símbolos reenviados al codificador entrópico respectivo, hasta el símbolo b; (1.2) no se produce b dentro de los siguientes n-1 símbolos tras el primer símbolo, en cuyo caso el codificador entrópico respectivo está configurado para escribir una palabra de código en el flujo de bits respectivo, que es igual a $(b,...,b)_{n-1}$; o (2) el primer símbolo es igual a b, en cuyo caso el codificador entrópico respectivo está configurado para examinar los siguientes símbolos reenviados al codificador entrópico respectivo para determinar si (2.1) se produce a dentro de los siguientes n-2 símbolos tras el primer símbolo, en cuyo caso el codificador entrópico respectivo está configurado para escribir una palabra de código en el flujo de bits respectivo, que es igual al primer símbolo seguido por los siguientes símbolos reenviados al codificador entrópico respectivo hasta el símbolo a; o (2.2) no se produce a dentro de los siguientes n-2 símbolos tras el primer símbolo, en cuyo caso el codificador entrópico respectivo está configurado para escribir una palabra de código en el flujo de bits respectivo, que es igual a $(a,...,a)_n$.

Adicional o alternativamente, un segundo subconjunto de los codificadores 10 entrópicos puede ser un codificador de longitud variable configurado para mapear secuencias de símbolos de longitudes variables en palabras de código de longitudes fijas, respectivamente, usando cada uno de los codificadores entrópicos del segundo subconjunto una regla de mapeo biyectivo según la cual palabras de código de un código unario truncado primario con 2^n+1 palabras de código del tipo {(a), (ba), (bba), ... ,(b...ba), (bb...b)} con $b \neq a$ y $a,b \in \{0,1\}$ se mapean en palabras de código de un código libre de prefijo secundario de tal manera que la palabra de código (bb...b) del código unario truncado primario se mapean en la palabra de código (c) del código libre de prefijo secundario y todas las demás palabras de código {(a), (ba), (bba), ... ,(b...ba)} del código unario truncado primario se mapean en palabras de código que tienen (d) con $c \neq d$ y $c,d \in \{0,1\}$ como prefijo y una palabra de n bits como sufijo, en el que los codificadores entrópicos usan diferentes n. Cada uno del segundo subconjunto de codificadores entrópicos puede estar configurado de tal manera que la palabra de n bits es una representación de n bits del número de b en la palabra de código respectiva del código unario truncado primario. Sin embargo, son viables otras alternativas.

De nuevo, desde el punto de vista del modo de funcionamiento del codificador 10 respectivo, cada uno del segundo subconjunto de codificadores entrópicos puede estar configurado para, al convertir los símbolos reenviados al codificador entrópico respectivo en el flujo de bits respectivo, contar un número de b en una secuencia de símbolos reenviados al codificador entrópico respectivo, hasta que se produce una a, o hasta que el número de la secuencia de símbolos reenviados al codificador entrópico respectivo alcanza 2” siendo los 2” símbolos de la secuencia b, y (1) si el número de b es igual a 2^n, escribir c con $c \in \{0,1\}$ como palabra de código de un código libre de prefijo secundario en el flujo de bits respectivo, y (2) si el número de b es inferior a 2^n, escribir una palabra de código del código libre de prefijo secundario en el flujo de bits respectivo, que tiene (d) con $c \neq d$ y $d \in \{0,1\}$ como prefijo y una palabra de n bits determinada dependiendo del número de b como sufijo.

También adicional o alternativamente, uno predeterminado de los codificadores 10 entrópicos puede ser un codificador de longitud variable configurado para mapear secuencias de símbolos de longitudes fijas en palabras de código de longitudes variables, respectivamente, usando el codificador entrópico predeterminado una regla de mapeo biyectivo según la cual 2^3 palabras de código de longitud 3 de un código primario se mapean en palabras de código de un código libre de prefijo secundario de tal manera que la palabra de código $(aaa)_3$ del código primario con $a \in \{0,1\}$ se mapea en la palabra de código (c) con $c \in \{0,1\}$, las tres palabras de código del código primario que tienen exactamente una b con $b \neq a$ y $b \in \{0,1\}$ se mapean palabras de código que tienen (d) con $c \neq d$ y $d \in \{0,1\}$ como prefijo y una primera palabra de 2 bits respectiva de un primer conjunto de palabras de 2 bits como sufijo, las tres palabras de código del código primario que tienen exactamente una a se mapean en palabras de código que tienen (d) como prefijo y una concatenación de una primera palabra de 2 bits que no es un elemento del primer conjunto y una segunda palabra de 2 bits de un segundo conjunto de palabras de 2 bits, como sufijo, y en la que la palabra de código $(bbb)_3$ se mapea en una palabra de código que tiene (d) como prefijo y una concatenación de la primera palabra de 2 bits que no es un elemento del primer conjunto y una segunda palabra de 2 bits que no es un elemento del segundo conjunto, como sufijo. La primera palabra de 2 bits de las palabras de código del código primario que tienen exactamente una b pueden ser una representación de 2 bits de una posición de la b en la palabra de código respectiva del código primario, y la segunda palabra de 2 bits de las palabras de código del código primario que tienen exactamente una a puede ser una representación de 2 bits de una posición de la a en la palabra de código respectiva del código primario. Sin embargo, son viables otras alternativas.

De nuevo, el predeterminado de los codificadores entrópicos puede estar configurado para, al convertir los símbolos

reenviados al codificador entrópico predeterminado en el flujo de bits respectivo, examinar los símbolos para el codificador entrópico predeterminado en tripletes en cuanto a si (1) el triplete consiste en a, en cuyo caso el codificador entrópico predeterminado está configurado para escribir la palabra de código (c) en el flujo de bits respectivo, (2) el triplete comprende exactamente una b, en cuyo caso el codificador entrópico predeterminado está configurado para escribir una palabra de código que tiene (d) como prefijo y una representación de 2 bits de una posición de la b en el triplete como sufijo, en el flujo de bits respectivo; (3) el triplete comprende exactamente una a, en cuyo caso el codificador entrópico predeterminado está configurado para escribir una palabra de código que tiene (d) como prefijo y una concatenación de la primera palabra de 2 bits que no es un elemento del primer conjunto y una representación de 2 bits de una posición de la a en el triplete como sufijo, en el flujo de bits respectivo; o (4) el triplete consiste en b, en cuyo caso el codificador entrópico predeterminado está configurado para escribir una palabra de código que tiene (d) como prefijo y una concatenación de la primera palabra de 2 bits que no es un elemento del primer conjunto y la primera palabra de 2 bits que no es un elemento del segundo conjunto como sufijo, en el flujo de bits respectivo.

Con respecto al lado de decodificación, las realizaciones que acaban de describirse dan a conocer un decodificador para reconstruir una secuencia de símbolos 326, que comprende una pluralidad de decodificadores 322 entrópicos, cada uno de los cuales está configurado para convertir un flujo 324 de bits respectivo en símbolos 321; un asignador 316 configurado para asignar varios parámetros a cada símbolo 326 de una secuencia de símbolos que va a reconstruirse basándose en información contenida dentro de símbolos anteriormente reconstruidos de la secuencia de símbolos; y un selector 318 configurado para recuperar cada símbolo 325 de la secuencia de símbolos que va a reconstruirse a partir de uno seleccionado de la pluralidad de decodificadores entrópicos, dependiendo la selección del número de parámetros definidos en el símbolo respectivo. Según las realizaciones que acaban de describirse al menos un primer subconjunto de los decodificadores 322 entrópicos son decodificadores de longitud variable configurados para mapear palabras de código de longitudes variables en secuencias de símbolos de longitudes variables, respectivamente, usando cada uno de los decodificadores entrópicos 22 del primer subconjunto una regla de mapeo biyectivo según la cual palabras de código de un código libre de prefijo primario con $(2n-1) \geq 3$ palabras de código se mapean en palabras de código de un código libre de prefijo secundario que es idéntico al código de prefijo primario de tal manera que todas salvo dos de las palabras de código del código libre de prefijo primario se mapean en palabras de código idénticas del código libre de prefijo secundario mientras que las dos palabras de código de los códigos libres de prefijo primario y secundario tienen longitudes diferentes y se mapean entre sí de una manera intercambiada, en la que los codificadores entrópicos usan diferentes n. El primer código libre de prefijo puede construirse de tal manera que las palabras de código del primer código libre de prefijo son $(a,b)_2$, $(a,a,b)_3$, ..., $(a,...,a,b)_n$, $(a,...,a)_n$, $(b,a)_2$, $(b,b,a)_3$, ..., $(b,...,b,a)_{n-1}$, $(b,...,b)_{n-1}$, y las dos palabras de código mapeadas entre sí de la manera intercambiada pueden ser $(a,...,a)_n$ y $(b,...,b)_{n-1}$ con $b \neq a$ y $a,b \in \{0,_1\}$. Sin embargo, son viables otras alternativas.

Cada uno del primer subconjunto de codificadores entrópicos puede estar configurado para, al convertir el flujo de bits respectivo en los símbolos, examinar un primer bit del flujo de bits respectivo, para determinar si (1) el primer bit es igual a a 0 {0,1}, en cuyo caso el codificador entrópico respectivo está configurado para examinar los siguientes bits del flujo de bits respectivo para determinar si (1.1) se produce b con $b \neq a$ y b 0 {0,1} dentro de los siguientes n-1 bits tras el primer bit, en cuyo caso el decodificador entrópico respectivo está configurado para reconstruir una secuencia de símbolos, que es igual al primer bit seguido por los siguientes bits del flujo de bits respectivo, hasta el bit b; o (1.2) no se produce b dentro de los siguientes n-1 bits tras el primer bit, en cuyo caso el decodificador entrópico respectivo está configurado para reconstruir una secuencia de símbolos, que es igual a $(b,...,b)_{n-1}$; o (2) el primer bit es igual a b, en cuyo caso el decodificador entrópico respectivo está configurado para examinar los siguientes bits del flujo de bits respectivo para determinar si (2.1) se produce a dentro de los siguientes n-2 bits tras el primer bit, en cuyo caso el decodificador entrópico respectivo está configurado para reconstruir una secuencia de símbolos, que es igual al primer bit seguido por los siguientes bits del flujo de bits respectivo hasta el símbolo a; o (2.2) no se produce a dentro de los siguientes n-2 bits tras el primer bit, en cuyo caso el decodificador entrópico respectivo está configurado para reconstruir una secuencia de símbolos, que es igual a $(a,...,a)_n$.

Adicional o alternativamente, al menos un segundo subconjunto de los decodificadores 322 entrópicos puede ser un decodificador de longitud variable configurado para mapear palabras de código de longitudes fijas en secuencias de símbolos de longitudes variables, respectivamente, usando cada uno de los decodificadores entrópicos del segundo subconjunto una regla de mapeo biyectivo según la cual palabras de código de un código libre de prefijo secundario se mapean en palabras de código de un código unario truncado primario con 2^n+1 palabras de código del tipo {(a), (ba), (bba), ... ,(b...ba), (bb...b)} con $b \neq a$ y $a,b \in \{0,1\}$ de tal manera que palabra de código (c) del código libre de prefijo secundario se mapea en la palabra de código (bb...b) del código unario truncado primario y palabras de código que tienen (d) con $c \neq d$ y $c,d \in \{0,1\}$ como prefijo y una palabra de n bits como sufijo, se mapean en una respectiva de las otras palabras de código {(a), (ba), (bba), ... ,(b...ba)} del código unario truncado primario, en la que los decodificadores entrópicos usan diferentes n. Cada uno del segundo subconjunto de decodificadores entrópicos puede estar configurado de tal manera que la palabra de n bits es una representación de n bits del número de b en la palabra de código respectiva del código unario truncado primario. Sin embargo, son viables otras alternativas.

Cada uno de un segundo subconjunto de decodificadores entrópicos puede ser un decodificador de longitud variable configurado para mapear palabras de código de longitudes fijas en secuencias de símbolos de longitudes variables, respectivamente, y configurado para, al convertir el flujo de bits del decodificador entrópico respectivo en los símbolos, examinar un primer bit del flujo de bits respectivo para determinar si (1) el mismo es igual a c con c ∈ {0,1}, en cuyo caso el decodificador entrópico respectivo está configurado para reconstruir una secuencia de símbolos que es igual a $(bb...b)_2^n$ con b ∈ {0,1}; o (2) el mismo es igual a d con c ≠ d y c,d ∈ {0,1}, en cuyo caso el decodificador entrópico respectivo está configurado para determinar una palabra de n bits a partir de n bits adicionales del flujo de bits respectivo, tras el primer bit, y reconstruir una secuencia de símbolos a partir de la misma que es del tipo {(a), (ba), (bba), ... ,(b...ba), (bb...b)} con b ≠ a y b ∈ {0,1} dependiendo el número de b de la palabra de n bits.

Adicional o alternativamente, uno predeterminado de los decodificadores 322 entrópicos puede ser un decodificador de longitud variable configurado para mapear palabras de código de longitudes variables en secuencias de símbolos de longitudes fijas, respectivamente, usando el decodificador entrópico predeterminado una regla de mapeo biyectivo según la cual palabras de código de un código libre de prefijo secundario se mapean en 2^3 palabras de código de longitud 3 de un código primario de tal manera que la palabra de código (c) con c ∈ {0,1} se mapea en la palabra de código $(aaa)_3$ del código primario con a ∈ {0,1}, las palabras de código que tienen (d) con c ≠ d y d ∈ {0,1} como prefijo y una primera palabra de 2 bits respectiva de un primer conjunto de tres palabras de 2 bits como sufijo se mapean en las tres palabras de código del código primario que tienen exactamente una b con b≠a y b ∈ {0,1}, las palabras de código que tienen (d) como prefijo y una concatenación de una primera palabra de 2 bits que no es un elemento del primer conjunto y una segunda palabra de 2 bits de un segundo conjunto de tres palabras de 2 bits, como sufijo se mapean en las tres palabras de código del código primario que tienen exactamente una a, y una palabra de código que tiene (d) como prefijo y una concatenación de la primera palabra de 2 bits que no es un elemento del primer conjunto y una segunda palabra de 2 bits que no es un elemento del segundo conjunto, como sufijo se mapea en la palabra de código $(bbb)_3$. La primera palabra de 2 bits de las palabras de código del código primario que tienen exactamente una b puede ser una representación de 2 bits de una posición de la b en la palabra de código respectiva del código primario, y la segunda palabra de 2 bits de las palabras de código del código primario que tienen exactamente una a puede ser una representación de 2 bits de una posición de la a en la palabra de código respectiva del código primario. Sin embargo, son viables otras alternativas.

El predeterminado de los decodificadores entrópicos puede ser un decodificador de longitud variable configurado para mapear palabras de código de longitudes variables en secuencias de símbolos de tres símbolos cada una, respectivamente, y configurado para, al convertir el flujo de bits del decodificador entrópico respectivo en los símbolos, examinar el primer bit del flujo de bits respectivo para determinar si (1) el primer bit del flujo de bits respectivo es igual a c con c ∈ {0,1}, en cuyo caso el decodificador entrópico predeterminado está configurado para reconstruir una secuencia de símbolos que es igual a $(aaa)_3$ con a 0 {0,1}, o (2) el primer bit del flujo de bits respectivo es igual a d con c ≠ d y d ∈ {0,1}, en cuyo caso el decodificador entrópico predeterminado está configurado para determinar una primera palabra de 2 bits a partir de 2 bits adicionales del flujo de bits respectivo, tras el primer bit, y examinar la primera palabra de 2 bits para determinar si (2.1) la primera palabra de 2 bits no es un elemento de un primer conjunto de tres palabras de 2 bits, en cuyo caso el decodificador entrópico predeterminado está configurado para reconstruir una secuencia de símbolos que tiene exactamente una b con b≠a y b 0 {0,1}, dependiendo la posición de b en la secuencia de símbolos respectiva de la primera palabra de 2 bits, o (2.2) la primera palabra de 2 bits es un elemento del primer conjunto, en cuyo caso el decodificador entrópico predeterminado está configurado para determinar una segunda palabra de 2 bits a partir de 2 bits adicionales del flujo de bits respectivo, tras los dos bits a partir de los cuales se ha determinado la primera palabra de 2 bits, y examinar la segunda palabra de 2 bits para determinar si (3.1) la segunda palabra de 2 bits no es un elemento de un segundo conjunto de tres palabras de 2 bits, en cuyo caso el decodificador entrópico predeterminado está configurado para reconstruir una secuencia de símbolos que tiene exactamente una a, dependiendo la posición de la a en la secuencia de símbolos respectiva de la segunda palabra de 2 bits, o (3.2) la segunda palabra de 2 bits es un elemento de un segundo conjunto de tres palabras de 2 bits, en cuyo caso el decodificador entrópico predeterminado está configurado para reconstruir una secuencia de símbolos que es igual a $(bbb)_3$.

Ahora, tras haber descrito el concepto general de un esquema de codificación de vídeo, se describen realizaciones de la presente invención con respecto a las realizaciones anteriores. Dicho de otro modo, las realizaciones explicadas resumidamente a continuación pueden implementarse mediante el uso de los esquemas anteriores, y viceversa, los esquemas de codificación anteriores pueden implementarse usando y aprovechando las realizaciones explicadas resumidamente a continuación.

En las realizaciones anteriores descritas con respecto a las figuras 7 a 9, el codificador entrópico y los decodificadores de las figuras 1 a 6 se implementaron según un concepto de PIPE. Una realización especial usó codificadores/decodificadores 310 y 322 de un único estado de probabilidad aritméticos. Tal como se describirá a continuación, según una realización alternativa, las entidades 306-310 y las entidades 318 a 322 correspondientes pueden sustituirse por un motor de codificación entrópica común. Como ejemplo, debe imaginarse un motor de codificación aritmética, que gestiona simplemente un estado común R y L y codifica todos los símbolos en un flujo de bits común, renunciando así a los aspectos ventajosos del presente concepto de PIPE con respecto al

procesamiento en paralelo, pero evitando la necesidad de entrelazar los flujos de bits parciales tal como se comenta adicionalmente a continuación. Al hacer esto, el número de estados de probabilidad mediante el cual se estiman las probabilidades del contexto mediante actualización (tal como consulta de tabla), puede ser superior al número de estados de probabilidad mediante el cual se realiza la subdivisión de intervalos de probabilidad. Es decir, de manera análoga a la cuantificación del valor de anchura de intervalo de probabilidad antes de indexar en la tabla Rtab, también puede cuantificarse el índice de estado de probabilidad. Por tanto, la descripción anterior de una posible implementación para los codificadores/decodificadores 310 y 322 individuales puede extenderse para un ejemplo de una implementación de los codificadores/decodificadores 318-322/306-310 entrópicos como motores de codificación/decodificación aritmética binarios adaptativos de contexto.

De manera más precisa, según una realización, el codificador entrópico acoplado a la salida del asignador de parámetros (que actúa como asignador de contextos, en este caso) pude funcionar de la siguiente manera:

0. El asignador 304 reenvía el valor de elemento binario junto con el parámetro de probabilidad. La probabilidad es pState_current[elemento binario].

1. Por tanto, el motor de codificación entrópica recibe: 1) valLPS, 2) el elemento binario y 3) la estimación de distribución de probabilidad pState_current[elemento binario]. pState_current[elemento binario] puede tener más estados que el número de índices de estado de probabilidad distinguibles de Rtab. Si es así, pState_current[elemento binario] puede cuantificarse tal como, por ejemplo, ignorando m LSB siendo m mayor de o igual a 1 y preferiblemente 2 ó 3 para obtener un p_state, es decir el índice que se usa después para acceder a la tabla Rtab. Sin embargo, la cuantificación puede omitirse, es decir p_state puede ser pState_current[elemento binario].

2. Después, se realiza una cuantificación de R (tal como se mencionó anteriormente: o bien se usa/gestiona un R (y L correspondiente con un flujo de bits común) para todos los valores distinguibles de p_state, o bien un R (y L correspondiente con flujo de bits parcial asociado por par de R/L) por valor distinguible de p_state, correspondiendo este último caso a tener un codificador 310 de elementos binarios por un valor de este tipo) q_index = Qtab[R>>q] (o alguna otra forma de cuantificación).

3. Después, se realiza una determinación de R_{LPS} y R:

R_{LPS} = Rtab[p_state][q_index]; Rtab tiene almacenado en el mismo valores previamente calculados para p[p_state]·Q[q_index]

R = R - R_{LPS} [es decir, R se actualiza previamente de manera preliminar como si “elemento binario” fuera MPS]

4. Cálculo del nuevo intervalo parcial:

si (elemento binario = 1 – valMPS) entonces

$$L \neg L + R$$

$$R \neg R_{LPS}$$

5. Renormalización de L y R, escritura de bits.

De manera análoga, el decodificador entrópico acoplado a la salida del asignador de parámetros (que actúa como asignador de contextos, en este caso) puede funcionar de la siguiente manera:

0. El asignador 304 reenvía el valor de elemento binario junto con el parámetro de probabilidad. La probabilidad es pState_current[elemento binario].

1. Por tanto, el motor de decodificación entrópica recibe la solicitud de un elemento binario junto con: 1) valLPS y 2) la estimación de distribución de probabilidad pState_current[elemento binario]. pState_current[elemento binario] puede tener más estados que el número de índices de estado de probabilidad distinguibles de Rtab. Si esto es así, pState_current[elemento binario] puede cuantificarse tal como, por ejemplo, ignorando m LSB siendo m mayor de o igual a 1 y preferiblemente 2 ó 3 para obtener un p_state, es decir el índice que se usa después para acceder a la tabla Rtab. Sin embargo, la cuantificación puede omitirse, es decir p_state puede ser pState_current[elemento binario].

2. Después, se realiza una cuantificación de R (tal como se mencionó anteriormente: o bien se usa/gestiona un R (y V correspondiente con un flujo de bits común) para todos los valores distinguibles de p_state, o bien un R (y V correspondiente con flujo de bits parcial asociado por par de R/L) por valor distinguible de p_state, correspondiendo este último caso a tener un codificador 310 de elementos binarios por un valor de este tipo)

q_index = Qtab[R>>q] (o alguna otra forma de cuantificación)

3. Después, se realiza una determinación de R_{LPS} y R:

R_{LPS} = Rtab[p_state][q_index]; Rtab tiene almacenado en el mismo valores previamente calculados para p[p_state]·Q[q_index]

R = R - R_{LPS} [es decir, R se actualiza previamente de manera preliminar como si “elemento binario” fuera MPS]

4. Determinación de elemento binario dependiendo de la posición del intervalo parcial:

si (V ^{3} R) entonces

elemento binario ¬ 1 – valMPS (el elemento binario se decodifica como LPS; el selector 18 de memoria intermedia de elemento binario obtendrá el valor de elemento binario real mediante el uso de esta información de elemento binario y valMPS)

V ¬ V – R

R ¬ R_{LPS}

si no

elemento binario ¬ val MPS (el elemento binario se decodifica como MPS; el valor de elemento binario real se obtiene usando esta información de elemento binario y valMPS)

5. Renormalización de R, lectura de un bit y actualización de V.

Tal como se describió anteriormente, el asignador 4 asigna pState_current[elemento binario] a cada elemento binario. La asociación puede realizarse basándose en una selección de contexto. Es decir, el asignador 4 puede seleccionar un contexto usando un índice de contexto ctxIdx que, a su vez, tiene una pState_current respectiva asociada con el mismo. Puede realizarse una actualización de probabilidad cada vez que se ha aplicado una probabilidad pState_current[elemento binario] a un elemento binario actual. Se realiza una actualización del estado de probabilidad pState_current[elemento binario] dependiendo del valor del bit codificado:

si (bit = 1 - valMPS) entonces

pState_current ← Next_State_LPS [pState_current]

si (pState_current = 0) entonces valMPS ← 1 - valMPS

si no

pState_current ← Next_State_MPS [pState_current]

Si se proporciona más de un contexto, la adaptación se realiza en función del contexto, es decir se usa pState_current[ctxIdx] para la codificación y después se actualiza usando el valor de elemento binario actual (codificado o decodificado, respectivamente).

Tal como se explicará resumidamente con más detalle a continuación, según realizaciones descritas ahora, el codificador y decodificador puede implementarse opcionalmente para funcionar en diferentes modos, concretamente modo de baja complejidad (LC) y de alta eficiencia (HE). Esto se ilustra principalmente con respecto a la codificación de PIPE a continuación (mencionando después los modos de PIPE de LC y HE), pero la descripción de los detalles de ajuste a escala de la complejidad puede transferirse fácilmente a otras implementaciones de los motores de codificación/decodificación entrópica tales como la realización de usar un codificador/decodificador aritmético adaptativo de contexto común.

Según las realizaciones explicadas resumidamente a continuación, ambos modos de codificación entrópica pueden compartir

• la misma sintaxis y semántica (para la secuencia de elementos 301 y 327 de sintaxis, respectivamente)

• los mismos esquemas de binarización para todos los elementos de sintaxis (tal como se especifica actualmente

para CABAC) (es decir, los binarizadores pueden funcionar independientemente del modo activado)

• el uso de los mismos códigos de PIPE (es decir, los codificadores/decodificadores de elementos binarios pueden funcionar independientemente del modo activado)

• el uso de valores de inicialización de modelo de probabilidad de 8 bits (en lugar de valores de inicialización de 16 bits tal como se especifica actualmente para CABAC)

De manera general, LC-PIPE difiere de HE-PIPE en la complejidad de procesamiento, tal como la complejidad de seleccionar la trayectoria 312 de PIPE para cada elemento binario.

Por ejemplo, el modo de LC puede funcionar con las siguientes restricciones: para cada elemento binario (binIdx), puede haber exactamente un modelo de probabilidad, es decir, un ctxIdx. Es decir, no puede proporcionarse ninguna selección/adaptación de contexto en LC PIPE. Sin embargo, elementos de sintaxis específicos tales como los usados para la codificación de residuos pueden codificarse usando contextos, tal como se explica resumidamente de manera adicional a continuación. Además, todos los modelos de probabilidad pueden ser no adaptativos, es decir, todos los modelos pueden inicializarse al comienzo de cada segmento con probabilidades de modelo apropiadas (dependiendo de la elección de tipo de segmento y QP de segmento) y pueden mantenerse fijas a lo largo de todo el procesamiento del segmento. Por ejemplo, pueden soportarse solo 8 probabilidades de modelo diferentes correspondientes a 8 códigos 310/322 de PIPE diferentes, tanto para modelado de contextos como para la codificación. Elementos de sintaxis específicos para codificación de residuos, es decir, significance_coeff_flag y coeff_abs_level_greaterX (con X = 1, 2), cuya semántica se explica resumidamente con más detalle a continuación, pueden asignarse a modelos de probabilidad de tal manera que (al menos) grupos de, por ejemplo, 4 elementos de sintaxis se codifican/decodifican con la misma probabilidad de modelo. En comparación con CAVLC, el modo LC-PIPE logra aproximadamente el mismo rendimiento de R-D y la misma producción.

HE-PIPE pude configurarse para ser conceptualmente similar a CABAC de H.264 con las siguientes diferencias: se sustituye la codificación aritmética binaria (BAC) por codificación de PIPE (igual que en el caso de LC-PIPE). Cada modelo de probabilidad, es decir, cada ctxIdx, puede representarse por un pipeIdx y un refineIdx, en los que pipeIdx con valores en el intervalo entre 0...7 representa la probabilidad de modelo de los 8 códigos de PIPE diferentes. Este cambio afecta solo a la representación interna de estados, no al comportamiento de la propia máquina de estados (es decir, estimación de probabilidad). Tal como se explicará resumidamente con más detalle a continuación, la inicialización de modelos de probabilidad puede usar valores de inicialización de 8 bits tal como se mencionó anteriormente. Puede usarse exploración hacia atrás de elementos de sintaxis coeff_abs_level_greaterX (con X = 1, 2), coeff_abs_level_minus3 y coeff_sign_flag (cuyas semánticas se aclararán a partir de la siguiente discusión) a lo largo de la misma trayectoria de exploración que la exploración hacia delante (usada, por ejemplo, en la codificación de mapa de significación). También puede simplificarse la derivación de contextos para la codificación de coeff_abs_level_greaterX (con X = 1, 2). En comparación con CABAC, el HE-PIPE propuesto logra aproximadamente el mismo rendimiento de R-D con una producción mejor.

Resulta fácil constatar que los modos que acaban de mencionarse se generan fácilmente haciendo, por ejemplo, que el motor de codificación/decodificación aritmética binaria adaptativa de contexto anteriormente mencionado sea de tal manera que el mismo funcione en diferentes modos.

Por tanto, según una realización según un primer aspecto de la presente invención, puede construirse un decodificador para decodificar un flujo de datos tal como se muestra en la figura 11. El decodificador es para decodificar un flujo 401 de datos, tal como un flujo 340 de bits entrelazado, en el que se codifican datos de medios, tales como datos de vídeo. El decodificador comprende un conmutador 400 de modo configurado para activar el modo de baja complejidad o el modo de alta eficiencia dependiendo del flujo 401 de datos. Para ello, el flujo 401 de datos puede comprender un elemento de sintaxis tal como un elemento de sintaxis binario, que tiene un valor binario de 1 en el caso de que el modo de baja complejidad sea el activado, y que tiene un valor binario de 0 en el caso de que el modo de alta eficiencia sea el activado. Evidentemente, la asociación entre valor binario y modo de codificación puede conmutarse, y también puede usarse un elemento de sintaxis no binario que tiene más de dos valores posibles. Dado que la selección real entre ambos modos aún no queda clara antes de la recepción del elemento de sintaxis respectivo, este elemento de sintaxis puede estar contenido dentro de alguna cabecera inicial del flujo 401 de datos codificado, por ejemplo, con una estimación de probabilidad o modelo de probabilidad fija o que se escribe en el flujo 401 de datos tal cual, es decir, usando un modo de derivación.

Además, el decodificador de la figura 11 comprende una pluralidad de decodificadores 322 entrópicos cada uno de los cuales está configurado para convertir palabras de código en el flujo 401 de datos en secuencias 321 parciales de símbolos. Tal como se describió anteriormente, puede conectarse un desentrelazador 404 entre entradas de decodificadores 322 entrópicos por un lado y la entrada del decodificador de la figura 11 en el que se aplica el flujo 401 de datos, por otro lado. Además, tal como ya se describió anteriormente, cada uno de los decodificadores 322 entrópicos puede asociarse con un intervalo de probabilidad respectivo, cubriendo los intervalos de probabilidad de

los diversos decodificadores entrópicos en conjunto el intervalo de probabilidad completo entre 0 y 1 (o entre 0 y 0,5 en el caso de los decodificadores 322 entrópicos que tratan con MPS y LPS en vez de valores de símbolo absolutos). Anteriormente se describieron detalles referentes a esta cuestión. Más adelante, se supone que el número de decodificadores 322 es 8 estando un índice de PIPE asociado con cada decodificador, pero cualquier otro número también es viable. Además, uno de estos codificadores, a continuación esto es a modo de ejemplo el que tiene pipe_id 0, está optimizado para elementos binarios que tienen estadísticas equiprobables, es decir su valor de elemento binario adopta 1 y 0 con igual probabilidad. Este decodificador puede pasar simplemente los elementos binarios. El codificador 310 respectivo funciona de la misma manera. Incluso puede omitirse cualquier manipulación de elementos binarios dependiendo del valor del valor de elemento binario más probable, valMPS, por los selectores 402 y 502, respectivamente. Dicho de otro modo, la entropía del flujo parcial respectivo ya es óptima.

Además, el decodificador de la figura 11 comprende un selector 402 configurado para recuperar cada símbolo de una secuencia 326 de símbolos a partir de uno seleccionado de la pluralidad de decodificadores 322 entrópicos. Tal como se mencionó anteriormente, el selector 402 puede estar dividido en un asignador 316 de parámetros y un selector 318. Un desimbolizador 314 está configurado para desimbolizar la secuencia 326 de símbolos con el fin de obtener una secuencia 327 de elementos de sintaxis. Un reconstructor 404 está configurado para reconstruir los datos 405 de medios basándose en la secuencia de elementos 327 de sintaxis. El selector 402 está configurado para realizar la selección dependiendo del modo activado del modo de baja complejidad y el modo de alta eficiencia tal como se indica mediante la flecha 406.

Tal como ya se indicó anteriormente, el reconstructor 404 puede ser parte de un decodificador de vídeo basado en bloques predictivo que funciona con una sintaxis y semántica fijas de elementos de sintaxis, es decir, fijas con respecto a la selección de modo mediante el conmutador 400 de modo. Es decir, la construcción del reconstructor 404 no experimenta problemas debido a la conmutabilidad de modo. Más precisamente, el reconstructor 404 no aumenta el coste de implementación debido a la conmutabilidad de modo ofrecida por el conmutador 400 de modo y al menos la funcionalidad con respecto a los datos de residuo y los datos de predicción siguen siendo iguales independientemente del modo seleccionado por el conmutador 400. Sin embargo, se aplica lo mismo con respecto a los decodificadores 322 entrópicos. Todos estos decodificadores 322 vuelven a usarse en ambos modos y, por consiguiente, no hay ningún coste de implementación adicional, aunque el decodificador de la figura 11 sea compatible con ambos modos, los modos de baja complejidad y de alta eficiencia.

Como aspecto secundario debe observarse que el decodificador de la figura 11 no solo puede funcionar con flujos de datos autocontenidos o bien en un modo o bien en el otro. En vez de eso, el decodificador de la figura 11 así como el flujo 401 de datos pueden estar configurados de tal manera que la conmutación entre ambos modos será posible incluso durante un fragmento de datos de medios tal como durante algún fragmento de audio o de vídeo, con el fin, por ejemplo, de controlar la complejidad de codificación en el lado de decodificación dependiendo de condiciones externas o del entorno tales como un estado de batería o similar usando un canal de realimentación desde el decodificador hasta el codificador con el fin de controlar en bucle cerrado en consecuencia la selección de modo.

Por tanto, el decodificador de la figura 11 funciona de manera similar en ambos casos, en el caso de seleccionarse el modo de LC o seleccionarse el modo de HE. El reconstructor 404 realiza la reconstrucción usando los elementos de sintaxis y solicita el elemento de sintaxis actual de un tipo de elemento de sintaxis predeterminado procesando u obedeciendo alguna recomendación de estructura de sintaxis. El desimbolizador 314 solicita varios elementos binarios con el fin de proporcionar una binarización válida para el elemento de sintaxis solicitado por el reconstructor 404. Evidentemente, en el caso de un alfabeto binario, la binarización realizada por el desimbolizador 314 se reduce a simplemente pasar el símbolo/elemento 326 binario respectivo al reconstructor 404 como el elemento de sintaxis binario actualmente solicitado.

Sin embargo, el selector 402 actúa independientemente en el modo seleccionado por el conmutador 400 de modo. El modo de funcionamiento del selector 402 tiende a ser más complejo en el caso del modo de alta eficiencia y menos complejo en el caso del modo de baja complejidad. Además, la siguiente discusión mostrará que el modo de funcionamiento del selector 402 en el modo menos complejo también tiende a reducir la tasa a la que el selector 402 cambia la selección entre los decodificadores 322 entrópicos en la recuperación de símbolos consecutivos a partir de los decodificadores 322 entrópicos. Dicho de otro modo, en el modo de baja complejidad, hay una probabilidad aumentada de que se recuperen símbolos inmediatamente consecutivos a partir del mismo decodificador entrópico entre la pluralidad de decodificadores 322 entrópicos. A su vez, esto permite una recuperación más rápida de los símbolos a partir de los decodificadores 322 entrópicos. A su vez, en el modo de alta eficiencia, el modo de funcionamiento del selector 402 tiende a conducir a una selección entre los decodificadores 322 entrópicos en la que el intervalo de probabilidad asociado con el decodificador 322 entrópico seleccionado respectivo se ajusta más estrechamente a las estadísticas de símbolo reales del símbolo actualmente recuperado por el selector 402, proporcionando así una mejor razón de compresión en el lado de codificación cuando se genera el flujo de datos respectivo según el modo de alta eficiencia.

Por ejemplo, el comportamiento diferente del selector 402 en ambos modos puede realizarse de la siguiente manera. Por ejemplo, el selector 402 puede estar configurado para realizar, para un símbolo predeterminado, la selección entre la pluralidad de decodificadores 322 entrópicos dependiendo de símbolos anteriormente recuperados de la secuencia 326 de símbolos en el caso de que esté activado el modo de alta eficiencia e independientemente de cualquier símbolo anteriormente recuperado de la secuencia de símbolos en el caso de que esté activado el modo de baja complejidad. La dependencia de símbolos anteriormente recuperados de la secuencia 326 de símbolos puede resultar de una adaptabilidad de contexto y/o una adaptabilidad de probabilidad. Ambas adaptabilidades pueden desactivarse durante el modo de baja complejidad en el selector 402.

Según una realización adicional, el flujo 401 de datos puede estar estructurado en porciones consecutivas tales como segmentos, tramas, grupo de imágenes, secuencias de tramas o similares, y cada símbolo de la secuencia de símbolos puede estar asociado con uno respectivo de una pluralidad de tipos de símbolo. En este caso, el selector 402 puede estar configurado para hacer variar, para símbolos de un tipo de símbolo predeterminado dentro de una porción actual, dependiendo la selección de símbolos anteriormente recuperados de la secuencia de símbolos del tipo de símbolo predeterminado dentro de la porción actual en el caso de que esté activado el modo de alta eficiencia, y dejar la selección constante dentro de la porción actual en el caso de que esté activado el modo de baja complejidad. Es decir, puede permitirse que el selector 402 cambie la selección entre los decodificadores 322 entrópicos para el tipo de símbolo predeterminado, pero estos cambios están restringidos a producirse entre transiciones entre porciones consecutivas. Mediante esta medida, las evaluaciones de estadísticas de símbolos reales están restringidas a instancias temporales que se producen con poca frecuencia mientras que la complejidad de codificación se reduce en la mayor parte del tiempo.

Además, cada símbolo de la secuencia 326 de símbolos puede estar asociado con uno respectivo de una pluralidad de tipos de símbolo, y el selector 402 puede estar configurado para seleccionar, para un símbolo predeterminado de un tipo de símbolo predeterminado, uno de una pluralidad de contextos que dependen de símbolos anteriormente recuperados de la secuencia 326 de símbolos y realizar la selección entre los decodificadores 322 entrópicos que depende de un modelo de probabilidad asociado con un contexto seleccionado junto con la actualización del modelo de probabilidad asociado con un contexto seleccionado que depende del símbolo predeterminado en el caso de que esté activado el modo de alta eficiencia, y realizar la selección de uno de la pluralidad de contextos que depende de los símbolos anteriormente recuperados de la secuencia 326 de símbolos y realizar la selección entre los decodificadores 322 entrópicos que depende del modelo de probabilidad asociado con el contexto seleccionado junto con dejar el modelo de probabilidad asociado con el contexto seleccionado constante en el caso de que esté activado el modo de baja complejidad. Es decir, el selector 402 puede usar la adaptabilidad de contexto con respecto a un determinado tipo de elemento de sintaxis en ambos modos, al tiempo que se suprime la adaptación de probabilidad en el caso del modo de LC.

Alternativamente, en lugar de suprimir completamente la adaptación de probabilidad, el selector 402 puede reducir simplemente una tasa de actualización de la adaptación de probabilidad del modo de LC con respecto al modo de HE.

Además, dicho de otro modo, posibles aspectos específicos de LC-PIPE, es decir, aspectos del modo de LC, pueden describirse de la siguiente manera. En particular, pueden usarse modelos de probabilidad no adaptativos en el modo de LC. Un modelo de probabilidad no adaptativo puede o bien tener una probabilidad preprogramada, es decir, en general constante o su probabilidad se mantiene fija en la totalidad del procesamiento de un segmento únicamente y por tanto puede establecerse dependiendo del tipo de segmento y QP, es decir, el parámetro de cuantificación que se indica, por ejemplo, dentro del flujo 401 de datos para cada segmento. Suponiendo que elementos binarios sucesivos asignados al mismo contexto siguen un modelo de probabilidad fija, es posible decodificar varios de esos elementos binarios en una etapa ya que se codificaron usando el mismo código de PIPE, es decir, usando el mismo decodificador entrópico, y se omite una actualización de probabilidad tras cada elemento binario decodificado. Al omitir actualizaciones de probabilidad se ahorran operaciones durante el proceso de codificación y decodificación y, por tanto, también conlleva reducciones de complejidad y una simplificación significativa en el diseño de hardware.

La restricción no adaptativa puede aliviarse para todos o algunos modelos de probabilidad seleccionados de tal manera que se permiten actualizaciones de probabilidad tras haberse codificado/decodificado un determinado número de elementos binarios usando este modelo. Un intervalo de actualización apropiado permite una adaptación de probabilidad al tiempo que tiene la capacidad de decodificar varios elementos binarios de una vez.

A continuación, se presenta una descripción más detallada de posibles aspectos comunes y de complejidad ajustable a escala de LC-PIPE y HE-PIPE. En particular, a continuación, se describen aspectos que pueden usarse para el modo de LC-PIPE y el modo de HE-PIPE de la misma manera o de una manera de complejidad ajustable a escala. Complejidad ajustable a escala significa que el caso de LC se deriva del caso de HE eliminando partes particulares o sustituyéndolas por algo menos complejo. Sin embargo, antes de proceder con el mismo, debe mencionarse que la realización de la figura 11 puede transferirse fácilmente a la realización de

codificación/decodificación aritmética binaria adaptativa de contexto anteriormente mencionada: el selector 402 y los decodificadores 322 entrópicos se condensarán en un decodificador aritmético binario adaptativo de contexto que recibirá el flujo 401 de datos directamente y seleccionará el contexto para un elemento binario que va a derivarse actualmente del flujo de datos. Esto es especialmente cierto para la adaptabilidad de contexto y/o adaptabilidad de probabilidad. Ambas funcionalidades/adaptabilidades pueden desactivarse, o diseñarse de manera más relajada, durante el modo de baja complejidad.

Por ejemplo, al implementar la realización de la figura 11, la etapa de codificación entrópica de PIPE que implica los decodificadores 322 entrópicos puede usar ocho códigos de variable a variable sistemáticos, es decir, cada decodificador 322 entrópico puede ser de un tipo v2v que se ha descrito anteriormente. El concepto de codificación de PIPE usando códigos v2v sistemáticos se simplifica restringiendo el número de códigos v2v. En el caso de un decodificador aritmético binario adaptativo de contexto, el mismo puede gestionar los mismos estados de probabilidad para los diferentes contextos y usar los mismos (o una versión cuantificada de los mismos) para la subdivisión de probabilidad. El mapeo de CABAC o estados de modelo de probabilidad, es decir los estados usados para la actualización de probabilidad, a ID de PIPE o índices de probabilidad para su consulta en Rtab puede ser tal como se representa en la tabla A.

Tabla A: Mapeo de estados de CABAC a índices de PIPE

Estado de CABAC	Índice de PIPE
0	0
1	
2	
3	1
4	
5	
6	
7	
8	
9	
10	2
11	
12	
13	
14	
15	3
16	
17	
18	
19	
20	
21	
22	4
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	5
33	
34	
35	
36	
37	
38	
39	
40	
41	

42	
43	
44	
45	
46	6
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	7

Este esquema de codificación modificado puede usarse como base para el enfoque de codificación de vídeo de complejidad ajustable a escala. Cuando se realiza adaptación de modo de probabilidad, el selector 402 o decodificador aritmético binario adaptativo de contexto, respectivamente, seleccionarán el decodificador 322 de PIPE, es decir derivarán el índice de PIPE, que va a usarse, y el índice de probabilidad en Rtab, respectivamente, basándose en el índice de estado de probabilidad (que oscila en este caso, a modo de ejemplo, entre 0 y 62) asociado con el símbolo que va a decodificarse actualmente (tal como mediante un contexto) usando el mapeo mostrado en la tabla A, y actualizarán este índice de estado de probabilidad dependiendo del símbolo actualmente decodificado usando, por ejemplo, valores de transición de recorrido de tabla específicos que apuntan al siguiente índice de estado de probabilidad que va a visitarse en el caso de un MPS y un LPS, respectivamente. En el caso de modo de LC, puede omitirse esta última actualización. Incluso puede omitirse el mapeo en el caso de modelos de probabilidad globalmente fijados.

Sin embargo, puede usarse una configuración de codificación entrópica arbitraria y también pueden usarse las técnicas en este documento con adaptaciones menores.

La descripción anterior de la figura 11 se refiere más bien de manera general a elementos de sintaxis y tipos de elemento de sintaxis. A continuación, se describe una codificación de complejidad configurable de niveles de coeficiente de transformación.

Por ejemplo, el reconstructor 404 puede estar configurado para reconstruir un bloque 200 de transformación de niveles 202 de coeficiente de transformación basándose en una porción de la secuencia de elementos de sintaxis independientemente de que esté activado el modo de alta eficiencia o el modo de baja complejidad, comprendiendo la porción de la secuencia 327 de elementos de sintaxis, de una manera no entrelazada, elementos de sintaxis de mapa de significación que definen un mapa de significación que indica posiciones de niveles de coeficiente de transformación distintos de cero dentro del bloque 200 de transformación, y después (seguido por) elementos de sintaxis de nivel que definen los niveles de coeficiente de transformación distintos de cero. En particular, pueden estar implicados los siguientes elementos: elementos de sintaxis de posición final (*last_significant_pos_x*, *last_significant_pos_y*) que indican una posición de un último nivel de coeficiente de transformación distinto de cero dentro del bloque de transformación; primeros elementos de sintaxis (*coeff_significant_flag*) que definen en conjunto un mapa de significación y que indican, para cada posición a lo largo de una trayectoria (274) unidimensional que conduce desde una posición de DC hasta la posición del último nivel de coeficiente de transformación distinto de cero dentro del bloque (200) de transformación, si el nivel de coeficiente de transformación en la posición respectiva es distinto de cero o no; segundos elementos de sintaxis (*coeff_abs_greater1*) que indican, para cada posición de la trayectoria (274) unidimensional en la que, según los primeros elementos de sintaxis binarios, está colocado un nivel de coeficiente de transformación distinto de cero, si el nivel de coeficiente de transformación en la posición respectiva es mayor de uno; y terceros elementos de sintaxis (*coeff_abs_greater2*, *coeff_abs_minus3*) que revelan, para cada posición de la trayectoria unidimensional en la que, según los primeros elementos de sintaxis binarios, está colocado un nivel de coeficiente de transformación mayor de uno, una cantidad en la que el nivel de coeficiente de transformación respectivo en la posición respectiva supera uno.

El orden entre los elementos de sintaxis de posición final, los primeros, los segundos y los terceros elementos de

sintaxis, puede ser el mismo para el modo de alta eficiencia y el modo de baja complejidad, y el selector 402 puede estar configurado para realizar la selección entre los decodificadores 322 entrópicos para símbolos a partir de los cuales el desimbolizador 314 obtiene los elementos de sintaxis de posición final, primeros elementos de sintaxis, segundos elementos de sintaxis y/o terceros elementos de sintaxis, dependiendo de manera diferente de que esté activado el modo de baja complejidad o el modo de alta eficiencia.

En particular, el selector 402 puede estar configurado para seleccionar, para símbolos de un tipo de símbolo predeterminado entre una subsecuencia de símbolos a partir de la cual el desimbolizador 314 obtiene los primeros elementos de sintaxis y segundos elementos de sintaxis, para cada símbolo del tipo de símbolo predeterminado uno de una pluralidad de contextos que dependen de símbolos anteriormente recuperados del tipo de símbolo predeterminado entre la subsecuencia de símbolos y realizar la selección dependiendo de un modelo de probabilidad asociado con el contexto seleccionado en el caso de que esté activado el modo de alta eficiencia, y realizar la selección de una manera constante por fragmentos de tal manera que la selección es constante a lo largo de subpartes continuas consecutivas de la subsecuencia en el caso de que esté activado el modo de baja complejidad. Tal como se describió anteriormente, las subpartes pueden medirse en el número de posiciones por las que se extiende la subparte respectiva cuando se mide a lo largo de la trayectoria 274 unidimensional, o en el número de elementos de sintaxis del tipo respectivo ya codificados con el contexto actual. Es decir, los elementos de sintaxis binarios *coeff_significant_flag*, *coeff_abs_greater1* y *coeff_abs_greater2*, por ejemplo, se codifican de manera adaptativa de contexto seleccionando el decodificador 322 basándose en el modelo de probabilidad del contexto seleccionado en el modo de HE. También se usa la adaptación de probabilidad. En modo de LC, también hay diferentes contextos que se usan para cada uno de los elementos de sintaxis binarios *coeff_significant_flag*, *coeff_abs_greater1* y *coeff_abs_greater2*. Sin embargo, para cada uno de estos elementos de sintaxis, el contexto se mantiene estático para la primera porción a lo largo de la trayectoria 274 cambiando el contexto simplemente en una transición a la siguiente porción que sigue inmediatamente a lo largo de la trayectoria 274. Por ejemplo, cada porción puede estar definida para tener 4, 8, 16 posiciones de bloque 200 de longitud, independientemente de si para la posición respectiva está presente el elemento de sintaxis respectivo o no. Por ejemplo, *coeff_abs_greater1* y *coeff_abs_greater2* simplemente están presentes para posiciones significativas, es decir posiciones en las que (o para las que) *coeff_significant_flag* es 1. Alternativamente, cada porción puede estar definida para tener 4, 8, 16 elementos de sintaxis de longitud, independientemente de si la porción respectiva así resultante se extiende a lo largo de un número mayor de posiciones de bloque. Por ejemplo, *coeff_abs_greater1* y *coeff_abs_greater2* simplemente están presentes para posiciones significativas, y, por tanto, porciones de cuatro elementos de sintaxis pueden extenderse cada una a lo largo de más de 4 posiciones de bloque debido a posiciones entre las mismas a lo largo de la trayectoria 274 para las que no se transmite ningún elemento de sintaxis de este tipo tal como ningún *coeff_abs_greater1* y *coeff_abs_greater2* porque el nivel respectivo en esta posición es cero.

El selector 402 puede estar configurado para seleccionar, para los símbolos del tipo de símbolo predeterminado entre la subsecuencia de símbolos a partir de la cual el desimbolizador obtiene los primeros elementos de sintaxis y segundos elementos de sintaxis, para cada símbolo del tipo de símbolo predeterminado el contexto de una pluralidad de contextos dependiendo de varios símbolos anteriormente recuperados del tipo de símbolo predeterminado dentro de la subsecuencia de símbolos, que tienen un valor de símbolo predeterminado y pertenecen a la misma subparte, o varios símbolos anteriormente recuperados del tipo de símbolo predeterminado dentro de la secuencia de símbolos, que pertenecen a la misma subparte. La primera alternativa ha sido cierta para *coeff_abs_greater1* y la alternativa secundaria ha sido cierta para *coeff_abs_greater2* según las realizaciones anteriores específicas.

Además, los terceros elementos de sintaxis que revelan, para cada posición de la trayectoria unidimensional en la que, según los primeros elementos de sintaxis binarios, está colocado un nivel de coeficiente de transformación mayor de uno, una cantidad en la que el nivel de coeficiente de transformación respectivo en la posición respectiva supera uno, pueden comprender elementos de sintaxis con valor de número entero, es decir *coeff_abs_minus3*, y el desimbolizador 314 puede estar configurado para usar una función de mapeo que puede controlarse mediante un parámetro de control para mapear un dominio de palabras de secuencia de símbolos en un codominio de los elementos de sintaxis con valor de número entero, y para establecer el parámetro de control por elemento de sintaxis con valor de número entero dependiendo de elementos de sintaxis con valor de número entero de terceros elementos de sintaxis anteriores si el modo de alta eficiencia está activado, y realizar el ajuste de una manera constante por fragmentos de tal manera que el ajuste es constante a lo largo de subpartes continuas consecutivas de la subsecuencia en el caso de que esté activado el modo de baja complejidad, en el que el selector 402 puede estar configurado para seleccionar uno predeterminado de los decodificadores (322) entrópicos para los símbolos de palabras de secuencia de símbolos mapeados en los elementos de sintaxis con valor de número entero, que está asociado con una distribución de probabilidad igual, tanto en el modo de alta eficiencia como en el modo de baja complejidad. Es decir, incluso el desimbolizador puede funcionar dependiendo del modo seleccionado por el conmutador 400 tal como se ilustra por la línea 407 discontinua. En lugar de un ajuste constante por fragmentos del parámetro de control, el desimbolizador 314 puede mantener el parámetro de control constante durante el segmento actual, por ejemplo, o constante de manera global a lo largo del tiempo.

A continuación, se describe un modelado de contextos de complejidad ajustable a escala.

La evaluación del mismo elemento de sintaxis del vecino de arriba e izquierdo para la derivación del índice de modelo de contexto es un enfoque común y se usa con frecuencia en el caso de HE, por ejemplo, para el elemento de sintaxis de diferencia de vector de movimiento. Sin embargo, esta evaluación requiere más almacenamiento en memoria intermedia y no permite la codificación directa del elemento de sintaxis. Además, para lograr un mayor rendimiento de codificación, pueden evaluarse más vecinos disponibles.

En una realización preferida, todos los elementos de sintaxis de evaluación de etapa de modelado de contextos de bloques cuadrados o rectangulares vecinos o unidades de predicción se fijan a un modelo de contexto. Esto es igual a deshabilitar la adaptabilidad en la etapa de selección de modelo de contexto. Para esta realización preferida, la selección de modelo de contexto que depende del índice de elemento binario de la cadena de elementos binarios tras la binarización no se modifica en comparación con el diseño actual para CABAC. En otra realización preferida, de manera adicional al modelo de contexto fijo para elementos de sintaxis empleado para la evaluación de vecinos, también se fija el modelo de contexto para el índice de elemento binario diferente. Obsérvese que la descripción no incluye la binarización y selección de modelo de contexto para la diferencia de vector de movimiento y los elementos de sintaxis relacionados con la codificación de los niveles de coeficiente de transformación.

En una realización preferida, solo se permite la evaluación del vecino izquierdo. Esto conduce a un almacenamiento en memoria intermedia reducido en la cadena de procesamiento porque el último bloque o línea de unidad de codificación ya no tiene que almacenarse. En una realización preferida adicional, solo se evalúan vecinos que se encuentran en la misma unidad de codificación.

En una realización preferida, se evalúan todos los vecinos disponibles. Por ejemplo, además del vecino de arriba e izquierdo, se evalúan el vecino de arriba a la izquierda, arriba a la derecha y abajo a la izquierda en caso de disponibilidad.

Es decir, el selector 402 de la figura 11 puede estar configurado para usar, para un símbolo predeterminado relacionado con un bloque predeterminado de los datos de medios, símbolos anteriormente recuperados de la secuencia de símbolos relacionados con un número superior de bloques vecinos diferentes de los datos de medios en el caso de que esté activado el modo de alta eficiencia con el fin de seleccionar uno de una pluralidad de contextos y realizar la selección entre los decodificadores 322 entrópicos dependiendo de un modelo de probabilidad asociado con el contexto seleccionado. Es decir, los bloques vecinos pueden ser vecinos en el dominio temporal y/o espacial. Pueden verse bloques vecinos en el espacio, por ejemplo, en las figuras 1 a 3. Después, el selector 402 puede ser sensible a la selección de modo mediante el conmutador 400 de modo para realizar una adaptación de contacto basándose en símbolos anteriormente recuperados o elementos de sintaxis relacionados con un número superior de bloques vecinos en el caso del modo de HE en comparación con el modo de LC reduciendo así el coste de almacenamiento tal como acaba de describirse.

A continuación, se describe una codificación de complejidad reducida de diferencias de vector de movimiento según una realización.

En la norma de códec de vídeo H.264/AVC, se transmite un vector de movimiento asociado con un macrobloque indicando la diferencia (diferencia de vector de movimiento - *mvd*) entre el vector de movimiento del macrobloque actual y la mediana del factor de predicción de vector de movimiento. Cuando se usa CABAC como codificador entrópico, la *mvd* se codifica de la siguiente manera. La *mvd* con valor de número entero se divide en una parte absoluta y una parte de signo. La parte absoluta se binariza usando una combinación de binarización unaria truncada y de Golomb exponencial de 3er orden, denominada prefijo y sufijo de la cadena de elementos binarios resultantes. Los elementos binarios relacionados con la binarización unaria truncada se codifican usando modelos de contexto, mientras que los elementos binarios relacionados con la binarización de Golomb exponencial se codifican en un modo de derivación, es decir con una probabilidad fijada de 0,5 con CABAC. La binarización unaria funciona de la siguiente manera. Sea n el valor de número entero absoluto de la *mvd*, entonces la cadena de elementos binarios resultante consiste en *n* veces “1” y un último “0”. Como ejemplo, sea *n* = 4, entonces la cadena de elementos binarios es “11110”. En el caso de binarización unaria truncada, existe un límite y si el valor supera este límite, la cadena de elementos binarios consiste en *n*+1 veces “1”. Para el caso de *mvd*, el límite es igual a 9. Esto significa que, si una *mvd* absoluta es igual a o mayor de 9, se codifica, dando como resultado 9 veces “1”, la cadena de elementos binarios consiste en un prefijo y un sufijo con binarización de Golomb exponencial. El modelado de contextos para la parte unaria truncada se realiza de la siguiente manera. Para el primer elemento binario de la cadena de elementos binarios, se toman los valores de *mvd* absoluta de los macrobloques vecinos de arriba e izquierdo si están disponibles (si no están disponibles, se deduce que el valor es 0). Si la suma para la componente específica (dirección horizontal o vertical) es mayor de 2, se selecciona el segundo modelo de contexto, si la suma absoluta es mayor de 32, se selecciona el tercer modelo de contexto, de lo contrario (la suma absoluta es menor de 3) se selecciona el primer modelo de contexto. Además, los modelos de contexto son diferentes para cada componente. Para el segundo elemento binario de la cadena de elementos binarios, se usa el cuarto modelo de

contexto y se emplea el quinto modelo de contexto para los elementos binarios restantes de la parte unaria. Cuando la *mvd* absoluta es igual a o mayor de 9, por ejemplo, todos los elementos binarios de la parte unaria truncada son iguales a “1”, se codifica la diferencia entre el valor de *mvd* absoluta y 9 en un modo de derivación con binarización de Golomb exponencial de 3er orden. En la última etapa, se codifica el signo de la *mvd* en un modo de derivación.

La última técnica de codificación para la *mvd* cuando se usa CABAC como codificador entrópico se especifica en el modelo de prueba actual (HM) del proyecto de codificación de vídeo de alta eficiencia (HEVC). En HEVC, los tamaños de bloque son variables y la forma especificada por un vector de movimiento se denomina unidad de predicción (PU). El tamaño de PU del vecino de arriba e izquierdo puede tener otras formas y tamaños distintos de la PU actual. Por tanto, siempre que sea relevante, la definición de vecino de arriba e izquierdo se denomina ahora vecino de arriba e izquierdo de la esquina superior izquierda de la PU actual. Para la propia codificación, solo puede cambiarse el proceso de derivación para el primer elemento binario según una realización. En lugar de evaluar la suma absoluta del MV a partir de los vecinos, puede evaluarse cada vecino por separado. Si el MV absoluto de un vecino está disponible y es mayor de 16, puede aumentarse el índice de modelo de contexto dando como resultado el mismo número de modelos de contexto para el primer elemento binario, mientras que la codificación del nivel de MVD absoluto restante y el signo son exactamente iguales que en H.264/AVC.

En la técnica explicada anteriormente de manera resumida sobre la codificación de la *mvd*, tienen que codificarse hasta 9 elementos binarios con un modelo de contexto, mientras que el valor restante de una *mvd* puede codificarse en un modo de derivación de baja complejidad junto con la información de signo. Esta presente realización describe una técnica para reducir el número de elementos binarios codificados con modelos de contexto dando como resultado un número aumentado de derivaciones y reduce el número de modelos de contexto requeridos para la codificación de *mvd*. Para eso, se reduce el valor de corte de 9 a 1 ó 2. Esto significa que solo se codifica el primer elemento binario que especifica si la *mvd* absoluta es mayor de cero usando el modelo de contexto o se codifica el primer y el segundo elementos binarios que especifican si la *mvd* absoluta es mayor de cero y uno usando el modelo de contexto, mientras que el valor restante se codifica en el modo de derivación y/o usando un código de VLC. Todos los elementos binarios resultantes de la binarización usando el código de VLC (no usando el código unario o unario truncado) se codifican usando un modo de derivación de baja complejidad. En el caso de PIPE, son posibles una inserción directa en y desde el flujo de bits. Además, puede usarse una definición diferente del vecino de arriba e izquierdo para derivar una mejor selección de modelo de contexto para el primer elemento binario, si es que se usa alguna.

En una realización preferida, se usan códigos de Golomb exponenciales para binarizar la parte restante de las componentes de MVD absoluta. Para ello, el orden del código de Golomb exponencial es variable. El orden del código de Golomb exponencial se deriva de la siguiente manera. Tras derivarse y codificarse el modelo de contexto para el primer elemento binario, y por tanto el índice de ese modelo de contexto, se usa el índice como orden para la parte de binarización de Golomb exponencial. En esta realización preferida, el modelo de contexto para el primer elemento binario oscila entre 1 - 3 dando como resultado el índice 0 - 2, que se usa como orden del código de Golomb exponencial. Esta realización preferida puede usarse para el caso de HE.

En una alternativa a la técnica explicada anteriormente de manera resumida de usar dos multiplicado por cinco contextos en la codificación de la MVD absoluta, con el fin de codificar los 9 elementos binarios de binarización de código unario, también pueden usarse 14 modelos de contexto (7 para cada componente). Por ejemplo, aunque los elementos binarios primero y segundo de la parte unaria pueden codificarse con cuatro contextos diferentes tal como se describió anteriormente, puede usarse un quinto contexto para el tercer elemento binario y puede usarse un sexto contexto con respecto al cuarto elemento binario, mientras que los elementos binarios del quinto al noveno se codifican usando un séptimo contexto. Por tanto, en este caso se requerirán incluso 14 contextos, y simplemente el valor restante puede codificarse en un modo de derivación de baja complejidad. Una técnica para reducir el número de elementos binarios codificados con modelos de contexto que dan como resultado un aumento del número de derivaciones y reducir el número de modelos de contexto requeridos para la codificación de MVD, es reducir el valor de corte tal como, por ejemplo, de 9 a 1 ó 2. Esto significa que solo el primer elemento binario que especifica si la MVD absoluta es mayor de cero se codificará usando un modelo de contexto o el primer y el segundo elementos binarios que especifican si la MVD absoluta es mayor de cero y uno se codificarán usando un modelo de contexto respectivo, mientras que el valor restante se codifica con un código de VLC. Todos los elementos binarios resultantes de la binarización usando el código de VLC se codifican usando un modo de derivación de baja complejidad. En el caso de PIPE, es posible una inserción directa en y desde el flujo de bits. Además, la realización presentada usa otra definición del vecino de arriba e izquierdo para derivar una mejor selección de modelo de contexto para el primer elemento binario. Además de esto, el modelado de contexto se modifica de tal manera que el número de modelos de contexto requeridos para el primer o el primer y el segundo elementos binarios se reduce conduciendo a una reducción de memoria adicional. Además, la evaluación de los vecinos tales como el vecino de arriba puede deshabilitarse dando como resultado el ahorro de la memoria/memoria intermedia de línea requerida para el almacenamiento de los valores de *mvd* de los vecinos. Finalmente, el orden de codificación de las componentes puede dividirse de manera que permite la codificación de los elementos binarios de prefijo para ambas componentes (es decir, elementos binarios codificados con modelos de contexto) seguido por la codificación de

elementos binarios de derivación.

En una realización preferida, se usan códigos de Golomb exponenciales para binarizar la parte restante de las componentes de *mvd* absoluta. Para ello, el orden del código de Golomb exponencial es variable. El orden del código de Golomb exponencial puede derivarse de la siguiente manera. Tras derivarse el modelo de contexto para el primer elemento binario, y por tanto el índice de ese modelo de contexto, se usa el índice como orden para la binarización de Golomb exponencial. En esta realización preferida, el modelo de contexto para el primer elemento binario oscila entre 1 - 3 dando como resultado el índice 0 - 2, que se usa como orden del código de Golomb exponencial. Esta realización preferida puede usarse para el caso de HE y el número de modelos de contexto se reduce hasta 6. Con el fin de reducir de nuevo el número de modelos de contexto y por tanto ahorrar memoria, las componentes horizontal y vertical pueden compartir los mismos modelos de contexto en una realización preferida adicional. En este caso, solo se requieren 3 modelos de contexto. Además, puede tenerse en cuenta solo el vecino izquierdo para la evaluación en una realización preferida adicional de la invención. En esta realización preferida, el umbral puede no estar modificado (por ejemplo, solo se usa un único umbral de 16 dando como resultado un parámetro de Golomb exponencial de 0 ó 1 o un único umbral de 32 dando como resultado un parámetro de Golomb exponencial de 0 ó 2). Esta realización preferida ahorra la memoria intermedia de línea requerida para el almacenamiento de *mvd*. En otra realización preferida, el umbral se modifica y es igual a 2 y 16. Para esta realización preferida, en total se requieren 3 modelos de contexto para la codificación de la *mvd* y el parámetro de Golomb exponencial posible oscila entre 0 - 2. En una realización preferida adicional, el umbral es igual a 16 y 32. De nuevo, la realización descrita es adecuada para el caso de HE.

En una realización preferida adicional de la invención, se reduce el valor de corte de 9 a 2. En esta realización preferida, el primer elemento binario y el segundo elemento binario pueden codificarse usando modelos de contexto. La selección de modelo de contexto para el primer elemento binario puede realizarse como en el estado de la técnica o modificarse de una manera descrita en la realización preferida anterior. Para el segundo elemento binario, se selecciona un modelo de contexto independiente como en el estado de la técnica. En una realización preferida adicional, el modelo de contexto para el segundo elemento binario se selecciona evaluando la *mvd* del vecino izquierdo. Para este caso, el índice de modelo de contexto es el mismo que para el primer elemento binario, mientras que los modelos de contexto disponibles son diferentes de los del primer elemento binario. En total, se requieren 6 modelos de contexto (obsérvese que las componentes comparten los modelos de contexto). De nuevo, el parámetro de Golomb exponencial puede depender del índice de modelo de contexto seleccionado del primer elemento binario. En otra realización preferida de la invención, el parámetro de Golomb exponencial depende del índice de modelo de contexto del segundo elemento binario. Las realizaciones descritas de la invención pueden usarse para el caso de HE.

En una realización preferida adicional de la invención, los modelos de contexto para ambos elementos binarios son fijos y no se derivan evaluando ninguno de los vecinos izquierdo o de arriba. Para esta realización preferida, el número total de modelos de contexto es igual a 2. En una realización preferida adicional de la invención, el primer elemento binario y el segundo elemento binario comparten el mismo modelo de contexto. Como resultado, solo se requiere un modelo de contexto para la codificación de la *mvd*. En ambas realizaciones preferidas de la invención, el parámetro de Golomb exponencial puede ser fijo e igual a 1. La realización preferida descrita de la invención es adecuada para la configuración tanto de HE como de LC.

En otra realización preferida, el orden de la parte de Golomb exponencial se deriva independientemente del índice de modelo de contexto del primer elemento binario. En este caso, se usa la suma absoluta de la selección de modelo de contexto habitual de H.264/AVC para derivar el orden para la parte de Golomb exponencial. Esta realización preferida puede usarse para el caso de HE.

En una realización preferida adicional, el orden de los códigos de Golomb exponenciales es fijo y se establece a 0. En otra realización preferida, el orden de los códigos de Golomb exponenciales se fija y se establece a 1. En una realización preferida, el orden de los códigos de Golomb exponenciales se fija a 2. En una realización adicional, el orden de los códigos de Golomb exponenciales se fija a 3. En una realización adicional, el orden de los códigos de Golomb exponenciales se fija según la forma y el tamaño de la PU actual. Las realizaciones preferidas presentadas pueden usarse para el caso de LC. Obsérvese que el orden fijado de la parte de Golomb exponencial se considera con un número reducido de elementos binarios codificados con modelos de contexto.

En una realización preferida, los vecinos se definen de la siguiente manera. Para la PU anterior, se tienen en cuenta todas las PU que cubren la PU actual y se usa la PU con el MV más grande. Esto también se realiza para el vecino izquierdo. Se evalúan todas las PU que cubren la PU actual y se usa la PU con el MV más grande. En otra realización preferida, se usa el valor de vector de movimiento absoluto promedio de todas las PU que cubren el borde superior y el izquierdo de la PU actual para derivar el primer elemento binario.

Para las realizaciones preferidas presentadas anteriormente, es posible cambiar el orden de codificación de la siguiente manera. Tiene que especificarse la mvd para la dirección horizontal y vertical una tras otra (o viceversa).

Por tanto, tienen que codificarse dos cadenas de elementos binarios. Con el fin de minimizar el número de conmutaciones de modo para el motor de codificación entrópica (es decir, la conmutación entre el modo de derivación y el regular), es posible codificar los elementos binarios codificados con modelos de contexto para ambas componentes en la primera etapa seguido por los elementos binarios codificados en modo de derivación en la segunda etapa. Obsérvese que esto solo es una reordenación.

Obsérvese que los elementos binarios resultantes de la binarización unaria o unaria truncada también pueden representarse mediante una binarización de longitud fija equivalente de un indicador por índice de elemento binario que especifica si el valor es mayor que el índice de elemento binario actual. Como ejemplo, el valor de corte para la binarización unaria truncada de *mvd* se establece a 2 dando como resultado las palabras de código 0, 10, 11 para los valores 0, 1, 2. En la binarización de longitud fija correspondiente con un indicador por índice de elemento binario, un indicador para el índice de elemento binario 0 (es decir el primer elemento binario) especifica si el valor de *mvd* absoluto es mayor de 0 o no y un indicador para el segundo elemento binario con índice de elemento binario 1 especifica si el valor de *mvd* absoluto es mayor de 1 o no. Cuando el segundo indicador solo se codifica cuando el primer indicador es igual a 1, esto da como resultado las mismas palabras de código 0, 10, 11.

A continuación, se describe la representación de complejidad ajustable a escala del estado interno de modelos de probabilidad según una realización.

En la configuración de HE-PIPE, el estado interno de un modelo de probabilidad se actualiza tras codificar un elemento binario con el mismo. El estado actualizado se deriva mediante una consulta de tabla de transiciones de estado usando el estado antiguo y el valor del elemento binario codificado. En el caso de CABAC, un modelo de probabilidad puede adoptar 63 estados diferentes en los que cada estado corresponde a una probabilidad de modelo en el intervalo (0,0, 0,5). Cada uno de estos estados se usa para realizar dos probabilidades de modelo. Además de la probabilidad asignada al estado, también se usa 1,0 menos la probabilidad y un indicador denominado valMps almacena la información de si se usa la probabilidad o 1,0 menos la probabilidad. Esto conduce a un total de 126 estados. Para usar un modelo de probabilidad de este tipo con el concepto de codificación de PIPE, se necesita mapear cada uno de los 126 estados en uno de los codificadores de PIPE disponibles. En implementaciones actuales de codificadores de PIPE, esto se realiza usando una tabla de consulta. Un ejemplo de un mapeo de este tipo se representa en la tabla A.

A continuación, se describe una realización de cómo puede representarse el estado interno de un modelo de probabilidad para evitar usar una tabla de consulta para convertir el estado interno en un índice de PIPE. Únicamente se necesitan algunas operaciones de enmascaramiento de bits sencillas para extraer el índice de PIPE a partir de la variable de estado interno del modelo de probabilidad. Esta representación de complejidad ajustable a escala novedosa del estado interno de un modelo de probabilidad se diseña de una manera en dos niveles. Para aplicaciones en las que es obligatorio un funcionamiento de baja complejidad solo se usa el primer nivel. Solo describe el índice de PIPE y el indicador valMps que se usa para codificar o decodificar los elementos binarios asociados. En el caso del esquema de codificación entrópica de PIPE descrito, puede usarse el primer nivel para distinguir entre 8 probabilidades de modelo diferentes. Por tanto, el primer nivel necesitará 3 bits para pipeIdx y un bit adicional para el indicador valMps. Con el segundo nivel cada uno de los intervalos de probabilidad gruesos del primer nivel se refina para dar varios intervalos más pequeños que soportan la presentación de probabilidades a resoluciones superiores. Esta presentación más detallada permite un funcionamiento más exacto de estimadores de la probabilidad. En general, resulta adecuado para aplicaciones de codificación orientadas hacia altos rendimientos de RD. Como ejemplo esta representación de complejidad ajustable a escala del estado interno de modelos de probabilidad con el uso de PIPE se ilustra de la siguiente manera:

Primer nivel				Segundo nivel			
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
MPS	Idx de PIPE (0-7)			Idx de refinamiento (0-15)			

Los niveles primero y segundo se almacenan en una única memoria de 8 bits. Se requieren 4 bits para almacenar el primer nivel (un índice que define el índice de PIPE con el valor del MPS en el bit más significativo) y se usan otros 4 bits para almacenar el segundo nivel. Para implementar el comportamiento del estimador de probabilidad de CABAC, cada índice de PIPE tiene un número particular de índices de refinamiento permitidos que dependen de cuántos estados de CABAC se mapearon en el índice de PIPE. Por ejemplo, para el mapeo en la tabla A, el número de estados de CABAC por índice de PIPE se representa en la tabla B.

Tabla B: Número de estados de CABAC por índice de PIPE para el ejemplo de la tabla A.

Idx de PIPE	0	1	2	3	4	5	6	7
Número de estados de CABAC	3	7	5	7	10	14	16	1

Durante el proceso de codificación o decodificación de un elemento binario puede accederse al índice de PIPE y

valMps directamente empleando operaciones sencillas de desplazamiento de bits o enmascaramiento de bits. Los procesos de codificación de baja complejidad requieren únicamente los 4 bits del primer nivel y los procesos de codificación de alta eficiencia pueden usar adicionalmente los 4 bits del segundo nivel para realizar la actualización de modelo de probabilidad del estimador de probabilidad de CABAC. Para llevar a cabo esta actualización, puede diseñarse una tabla de consulta de transiciones de estado que realiza las mismas transiciones de estado que la tabla original, pero usando la representación de estados de dos niveles de complejidad ajustable a escala. La tabla de transiciones de estado original consiste en dos multiplicado por 63 elementos. Para cada estado de entrada, contiene dos estados de salida. Cuando se usa la representación de complejidad ajustable a escala, el tamaño de la tabla de transiciones de estado no supera dos multiplicado por 128 elementos, lo cual es un aumento aceptable del tamaño de tabla. Este aumento depende de cuántos bits se usan para representar el índice de refinamiento y para emular de manera exacta el comportamiento del estimador de probabilidad de CABAC, se necesitan cuatro bits. Sin embargo, puede usarse un estimador de probabilidad diferente, que puede funcionar con un conjunto reducido de estados de CABAC de tal manera que para cada índice de PIPE no se permiten más de 8 estados. Por tanto, puede hacerse coincidir el consumo de memoria con el nivel de complejidad dado del proceso de codificación adaptando el número de bits usados para representar el índice de refinamiento. En comparación con el estado interno de probabilidades de modelo con CABAC (en el que existen 64 índices de estado de probabilidad) se evita el uso de las consultas de tablas para mapear probabilidades de modelo en un código de PIPE específico y no se requiere ninguna conversión adicional.

A continuación, se describe una actualización de modelo de contexto de complejidad ajustable a escala según una realización.

Para actualizar un modelo de contexto, su índice de estado de probabilidad puede actualizarse basándose en uno o más elementos binarios anteriormente codificados. En la configuración de HE-PIPE, esta actualización se realiza tras la codificación o decodificación de cada elemento binario. A la inversa, en la configuración de LC-PIPE, esta actualización puede no realizarse nunca.

Sin embargo, es posible realizar una actualización de modelos de contexto de una manera de complejidad ajustable a escala. Es decir, la decisión de si actualizar un modelo de contexto o no puede basarse en diversos aspectos. Por ejemplo, una configuración de codificador puede no realizar actualizaciones únicamente para modelos de contexto particulares tales como, por ejemplo, los modelos de contexto de elemento de sintaxis *coeff_significant_flag*, y realizar siempre actualizaciones para todos los demás modelos de contexto.

Dicho de otro modo, el selector 402 puede estar configurado para realizar, para símbolos de cada uno de varios tipos de símbolo predeterminados, la selección entre los decodificadores 322 entrópicos dependiendo de un modelo de probabilidad respectivo asociado con el símbolo predeterminado respectivo de tal manera que el número de tipos de símbolo predeterminados es menor en el modo de baja complejidad que en comparación con el modo de alta eficiencia

Además, criterios para controlar si actualizar o no un modelo de contexto pueden ser, por ejemplo, el tamaño de un paquete de flujo de bits, el número de elementos binarios decodificados hasta ese momento, o la actualización se realiza únicamente tras codificar un número fijo o variable particular de elementos binarios para un modelo de contexto.

Con este esquema para decidir si actualizar modelos de contexto o no, puede implementarse actualización de modelo de contexto de complejidad ajustable a escala. Esto permite aumentar o reducir la porción de elementos binarios en un flujo de bits para el que se realizan actualizaciones de modelo de contexto. Cuanto mayor es el número de actualizaciones de modelo de contexto, mejor es la eficiencia de codificación y mayor es la complejidad computacional. Por tanto, la actualización de modelo de contexto de complejidad ajustable a escala puede lograrse con el esquema descrito.

En una realización preferida, la actualización de modelo de contexto se realiza para elementos binarios de todos los elementos de sintaxis excepto los elementos de sintaxis *coeff_significant_flag*, *coeff_abs_greater1* y *coeff_abs_greater2.*

En una realización preferida adicional, la actualización de modelo de contexto se realiza únicamente para elementos binarios de los elementos de sintaxis *coeff_significant_flag*, *coeff_abs_greater1* y *coeff_abs_greater2*.

En una realización preferida adicional, la actualización de modelo de contexto se realiza para todos los modelos de contexto cuando comienza la codificación o decodificación de un segmento. Tras procesarse un número predefinido particular de bloques de transformación, se deshabilita la actualización de modelo de contexto para todos los modelos de contexto hasta que se alcanza el final del segmento.

Por ejemplo, el selector 402 puede estar configurado para realizar, para símbolos de un tipo de símbolo

predeterminado, la selección entre los decodificadores 322 entrópicos dependiendo de un modelo de probabilidad asociado con el tipo de símbolo predeterminado junto con, o sin, la actualización del modelo de probabilidad asociado, de tal manera que la longitud de una fase de aprendizaje de la secuencia de símbolos a lo largo de la cual se realiza la selección de los símbolos del tipo de símbolo predeterminado junto con la actualización es más corta en el modo de baja complejidad en comparación con el modo de alta eficiencia.

Una realización preferida adicional es idéntica a la realización preferida anteriormente descrita, pero usa la representación de complejidad ajustable a escala del estado interno de modelos de contexto de una manera tal que una tabla almacena la “primera parte” (valMps y pipeIdx) de todos los modelos de contexto y una segunda tabla almacena la “segunda parte” (refineIdx) de todos los modelos de contexto. En el punto en el que se deshabilita la actualización de modelo de contexto para todos los modelos de contexto (tal como se describió en la realización preferida anterior), ya no se necesita la tabla que almacena la “segunda parte” y puede desecharse.

A continuación, se describe la actualización de modelo de contexto para una secuencia de elementos binarios según una realización.

En la configuración de LC-PIPE, los elementos binarios de elementos de sintaxis de tipo *coeff_significant_flag*, *coeff_abs_greater1* y *coeff_abs_greater2* se agrupan en subconjuntos. Para cada subconjunto, se usa un único modelo de contexto para codificar sus elementos binarios. En este caso, puede realizarse una actualización de modelo de contexto tras la codificación de un número fijo de elementos binarios de esta secuencia. Esto se denomina a continuación actualización de múltiples elementos binarios. Sin embargo, esta actualización puede diferir de la actualización que solo usa el último elemento binario codificado y el estado interno del modelo de contexto. Por ejemplo, para cada elemento binario que se codificó, se lleva a cabo una etapa de actualización de modelo de contexto.

A continuación, se facilitan ejemplos para la codificación de un subconjunto a modo de ejemplo que consiste en 8 elementos binarios. La letra “b” indica la decodificación de un elemento binario y la letra “u” indica la actualización del modelo de contexto. En el caso de LC-PIPE solo se realiza la decodificación de elementos binarios sin realizar actualizaciones de modelo de contexto:

bbbbbbbb

En el caso de HE-PIPE, tras la decodificación de cada elemento binario, se realiza una actualización de modelo de contexto:

bubububububububu

Con el fin de reducir en cierta medida la complejidad, la actualización de modelo de contexto puede realizarse tras una secuencia de elementos binarios (en este ejemplo, tras cada 4 elementos binarios se realizan las actualizaciones de esos 4 elementos binarios):

bbbbuuuubbbbuuuu

Es decir, el selector 402 puede estar configurado para realizar, para símbolos de un tipo de símbolo predeterminado, la selección entre los decodificadores 322 entrópicos dependiendo de un modelo de probabilidad asociado con el tipo de símbolo predeterminado junto con o sin la actualización del modelo de probabilidad asociado de tal manera que una frecuencia a la que se realiza la selección de los símbolos del tipo de símbolo predeterminado junto con la actualización es menor en el modo de baja complejidad que en comparación con el modo de alta eficiencia.

En este caso, tras la decodificación de 4 elementos binarios, siguen 4 etapas de actualización basadas en los 4 elementos binarios que acaban de decodificarse. Obsérvese que estas cuatro etapas de actualización pueden llevarse a cabo en una única etapa usando una tabla de consulta especial de consulta. Esta tabla de consulta almacena para cada combinación posible de 4 elementos binarios y cada estado interno posible del modelo de contexto el nuevo estado resultante tras las cuatro etapas de actualización convencionales.

En un modo determinado, se usa la actualización de múltiples elementos binarios para el elemento de sintaxis *coeff_significant_flag*. Para elementos binarios de todos los demás elementos de sintaxis, no se usa ninguna actualización de modelo de contexto. El número de elementos binarios que se codifican antes de realizarse una etapa de actualización de múltiples elementos binarios se establece a n. Cuando el número de elementos binarios del conjunto no puede dividirse entre n, quedan de 1 a n-1 elementos binarios al final del subconjunto tras la última actualización de múltiples elementos binarios. Para cada uno de estos elementos binarios, se realiza una actualización convencional de elementos binarios individuales tras codificar todos estos elementos binarios. El número n puede ser cualquier número positivo mayor de 1. Otro modo puede ser idéntico al modo anterior, excepto porque la actualización de múltiples elementos binarios se realiza para combinaciones arbitrarias de

coeff_significant_flag, *coeff_abs_greater1* y *coeff_abs_greater2* (en lugar de tan solo para *coeff_significant_flag*). Por tanto, este modo será más complejo que el otro. Todos los demás elementos de sintaxis (en los que no se usa la actualización de múltiples elementos binarios) pueden dividirse en dos subconjuntos independientes en los que para uno de los subconjuntos se usa actualización de elementos binarios individuales y para el otro subconjunto no se usa ninguna actualización de modelo de contexto. Cualquier subconjunto independiente posible es válido (incluyendo el subconjunto vacío).

En una realización alternativa, la actualización de múltiples elementos binarios puede basarse solo en los últimos m elementos binarios que se codifican inmediatamente antes de la etapa de actualización de múltiples elementos binarios. m puede ser cualquier número natural menor que n. Por tanto, la decodificación puede realizarse de la siguiente manera:

bbbbuubbbbuubbbbuubbbb...

con n=4 y m=2.

Es decir, el selector 402 puede estar configurado para realizar, para símbolos de un tipo de símbolo predeterminado, la selección entre los decodificadores 322 entrópicos dependiendo de un modelo de probabilidad asociado con el tipo de símbolo predeterminado, junto con la actualización del modelo de probabilidad asociado cada n-ésimo símbolo del tipo predeterminado basándose en los m símbolos más recientes del tipo de símbolo predeterminado de tal manera que la razón n/m es mayor en el modo de baja complejidad en comparación con el modo de alta eficiencia.

En una realización preferida adicional, para el elemento de sintaxis *coeff_significant_flag*, el esquema de modelado de contexto que usa una plantilla local tal como se describió anteriormente para la configuración de HE-PIPE puede usarse para asignar modelos de contexto a elementos binarios del elemento de sintaxis. Sin embargo, para estos elementos binarios, no se usa ninguna actualización de modelo de contexto.

Además, el selector 402 puede estar configurado para seleccionar, para símbolos de un tipo de símbolo predeterminado, uno de un número de contextos dependiendo de un número de símbolos anteriormente recuperados de la secuencia de símbolos y realizar la selección entre los decodificadores 322 entrópicos dependiendo de un modelo de probabilidad asociado con el contexto seleccionado, de tal manera que el número de contextos, y/o el número de símbolos anteriormente recuperados, es menor en el modo de baja complejidad en comparación con el modo de alta eficiencia.

Inicialización de modelo de probabilidad usando valores de inicialización de 8 bits

Esta sección describe el proceso de inicialización del estado interno de complejidad ajustable a escala de modelos de probabilidad usando un denominado valor de inicialización de 8 bits en lugar de dos valores de 8 bits tal como es el caso en la norma de codificación de vídeo del estado de la técnica H.265/AVC. Consiste en dos partes que son comparables a los pares de valores de inicialización usados para modelos de probabilidad en CABAC de H.264/AVC. Las dos partes representan los dos parámetros de una ecuación lineal para calcular el estado inicial de un modelo de probabilidad, que representa una probabilidad particular (por ejemplo, en forma de un índice de PIPE) a partir de un QP:

• La primera parte describe la pendiente y aprovecha la dependencia del estado interno con respecto al parámetro de cuantificación (QP) que se usa durante la codificación o decodificación.

• La segunda parte define un índice de PIPE a un QP dado así como el valMps.

Hay dos modos diferentes disponibles para inicializar un modelo de probabilidad usando el valor de inicialización dado. El primer modo se indica inicialización independiente de QP. Solo usa el índice de PIPE y valMps definido en la segunda parte del valor de inicialización para todos los QP. Esto es idéntico al caso en el que la pendiente es igual a 0. El segundo modo se indica inicialización dependiente de QP y usa adicionalmente la pendiente de la primera parte del valor de inicialización para alterar el índice de PIPE y para definir el índice de refinamiento. Las dos partes de un valor de inicialización de 8 bits se ilustran de la siguiente manera:

Primera parte				Segunda parte			
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Índice de pendiente				Índice de probabilidad de PIPE			

Consiste en dos partes de 4 bits. La primera parte contiene un índice que apunta a 1 de 16 pendientes predefinidas diferentes que se almacenan en una matriz. Las pendientes predefinidas consisten en 7 pendientes negativas (índice de pendiente 0-6), una pendiente que es igual a cero (índice de pendiente 7) y 8 pendientes positivas (índice

de pendiente 8-15). Las pendientes se representan en la tabla C.

Tabla C:

Índice de pendiente	0	1	2	3	4	5	6	7
Valor de pendiente	-239	-143	-85	-51	-31	-19	-11	0
Índice de pendiente	8	9	10	11	12	13	14	15
Valor de pendiente	11	19	31	51	85	143	239	399

Todos los valores se ajustan a escala mediante un factor de 256 para evitar el uso de operaciones de punto flotante. La segunda parte es el índice de PIPE que implementa la probabilidad creciente de valMps = 1 entre el intervalo de probabilidad p = 0 y p = 1. Dicho de otro modo, el codificador de PIPE n debe funcionar a una probabilidad de modelo superior al codificador de PIPE n - 1. Para cada modelo de probabilidad está disponible un índice de probabilidad de PIPE e identifica el codificador de PIPE cuyo intervalo de probabilidad contiene la probabilidad de $p_{valMPs=1}$ para QP = 26.

Tabla D: Mapeo de la segunda parte del valor de inicialización en codificadores de PIPE y valMps: UR = código de unario a Rice, TB = código de tres elementos binarios, BP = código de elemento binario-PIPE, EP = igual probabilidad (sin codificar)

Índice de probabilidad de PIPE	0	1	2	3	4	5	6	7
Codificador de PIPE	UR5	UR4	UR3	UR2	TB	BP2	BP3	EP
MPS	0	0	0	0	0	0	0	0
Índice de probabilidad de PIPE	8	9	10	11	12	13	14	15
Codificador de PIPE	EP	BP3	BP2	TB	UR2	UR3	UR4	UR5
MPS	1	1	1	1	1	1	1	1

Se requieren el QP y el valor de inicialización de 8 bits para calcular la inicialización del estado interno de los modelos de probabilidad calculando una simple ecuación lineal en forma de y = m * (QP - QPref) + 256 * b. Obsérvese que m define la pendiente que se toma de la tabla C usando el índice de pendiente (la primera parte del valor de inicialización de 8 bits) y b indica el codificador de PIPE a QPref = 26 (la segunda parte del valor de inicialización de 8 bits: “Índice de probabilidad de PIPE”). Entonces, valMPS es 1 y pipeIdx es igual a (y - 2048) >> 8 si y es mayor de 2047. De lo contrario, valMPS es 0 y pipeIdx es igual a (2047 - y)>> 8. El índice de refinamiento es igual a (((y-2048) & 255) * numStates) >> 8 si valMPS es igual a 1. De lo contrario, el índice de refinamiento es igual a (((2047-y) & 255) * numStates) >> 8. En ambos casos, numStates es igual al número de estados de CABAC de pipeIdx tal como se representa en la tabla B.

El esquema anterior puede usarse no solo en combinación con codificadores de PIPE, sino también en relación con los esquemas de CABAC anteriormente mencionados. En ausencia de PIPE, el número de estados de CABAC, es decir los estados de probabilidad entre los cuales se realiza la transición de estado en la actualización de probabilidad (pState_current[elemento binario]), por Idx de PIPE (es decir los bits más significativos respectivos de pState_current[elemento binario]) es entonces tan solo un conjunto de parámetros que realiza, de hecho, una interpolación lineal por fragmentos del estado de CABAC dependiendo del QP. Además, esta interpolación lineal por fragmentos también puede deshabilitarse prácticamente en el caso en el que el parámetro numStates usa el mismo valor para todos los Idx de PIPE. Por ejemplo, establecer numStates a 8 para todos los casos proporciona un total de 16 * 8 estados y el cálculo del índice de refinamiento se simplifica a ((y-2048) & 255)>> 5 para valMPS igual a 1 o ((2047-y)&255)>> 5 para valMPS igual a 0. Para este caso, mapear la representación usando valMPS, idx de PIPE, e idx de refinamiento de vuelta en la representación usada por el CABAC original de H.264/AVC resulta muy sencillo. El estado de CABAC viene dado como (Idx de PIPE << 3) + Idx de refinamiento. Este aspecto se describe adicionalmente a continuación con respecto a la figura 16.

A menos que la pendiente del valor de inicialización de 8 bits sea igual a cero o a menos que el QP sea igual a 26, es necesario calcular el estado interno empleando la ecuación lineal con el QP del proceso de codificación o decodificación. En el caso en el que la pendiente es igual a cero o que el QP del proceso de codificación actual es igual a 26, la segunda parte de valor de inicialización de 8 bits puede usarse directamente para inicializar el estado interno de un modelo de probabilidad. De lo contrario, la parte decimal del estado interno resultante puede aprovecharse adicionalmente para determinar un índice de refinamiento en aplicaciones de codificación de alta eficiencia mediante interpolación lineal entre los límites del codificador de PIPE específico. En esta realización preferida, la interpolación lineal se ejecuta multiplicando simplemente la parte decimal por el número total de índices de refinamiento disponibles para el codificador de PIPE actual y mapeando el resultado en el índice de refinamiento de número entero más próximo.

El proceso de inicialización del estado interno de los modelos de probabilidad puede hacerse variar con respecto al número de estados de índice de probabilidad de PIPE. En particular, la doble aparición del modo igualmente

probable usando el codificador de PIPE E1, es decir el uso de dos índices de PIPE diferentes para distinguir entre que MPS sea 1 ó 0, puede evitarse de la siguiente manera. De nuevo, el proceso puede invocarse durante el inicio del análisis sintáctico de los datos de segmento, y la entrada de este proceso puede ser un valor de inicialización de 8 bits tal como se representa en la tabla E, que se transmitirá, por ejemplo, dentro del flujo de bits para cada modelo de contexto que va a inicializarse.

Tabla E: Configuración de los 8 bits de initValue para un modelo de probabilidad

	Primeros 4 bits				Últimos 4 bits			
Bits de initValue	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Variable	slopeIdx				propIdx			

Los primeros 4 bits definen un índice de pendiente y se recuperan enmascarando los bits b4 - b7. Para cada índice de pendiente se especifica una pendiente (m) y se visualiza la tabla F.

Tabla F: Valores de variable m para slopeIdx

slopeIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
m	-239	-143	-85	-51	-31	-19	-11	0	11	19	31	51	85	143	239	399

Los bits b0-b3, los últimos 4 bits del valor de inicialización de 8 bits, identifican probIdx y describen la probabilidad a un QP predefinido. probIdx 0 indica la mayor probabilidad para símbolos con valor 0 y, respectivamente, probIdx 14 indica la mayor probabilidad para símbolos con valor 1. La tabla G muestra para cada probIdx el pipeCoder correspondiente y su valMps.

Tabla G: Mapeo de la parte de los últimos 4 bits del valor de inicialización en codificadores de PIPE y valMps: UR = código de unario a Rice, TB = código de tres elementos binarios, BP = código de elemento binario-PIPE, EP = igual probabilidad (sin codificar)

probIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
pipeCoder	UR5	UR4	UR3	UR2	TBC	BP2	BP3	EP	BP3	BP2	TBC	UR2	UR3	UR4	UR5
valMps	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Con ambos valores el cálculo del estado interno puede realizarse usando una ecuación lineal como y = m * x + 256 * b, donde m indica la pendiente, x indica el QP del segmento actual y b se deriva a partir de probIdx tal como se muestra en la siguiente descripción. Todos los valores en este proceso se ajustan a escala mediante un factor de 256 para evitar el uso de operaciones de punto flotante. La salida (y) de este proceso representa el estado interno del modelo de probabilidad al QP actual y se almacena en una memoria de 8 bits. Tal como se muestra en G, el estado interno consiste en los valMP, el pipeIdx y el refineIdx.

Tabla H: Configuración del estado interno de un modelo de probabilidad

	Primeros 4 bits				Últimos 4 bits			
Bits de initValue	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Variable	valMps	pipeIdx			refineIdx			

La asignación de refineIdx y pipeIdx es similar al estado interno de los modelos de probabilidad de CABAC (pStateCtx) y se presenta en H.

Tabla I: Asignación de pipeIdx, refineIdx y pStateCtx

pipeIdx	0			1							2					
refineIdx	0	1	2	0	1	2	3	4	5	6	0	1	2	3	4	
pStateCtx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
pipeIdx	3							4								
refineIdx	0	1	2	3	4	5	6	0	1	2	3	4	5	6	7	8	9
pStateCtx	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
pipeIdx	5															
refineIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13		
pStateCtx	32	33	34	35	36	37	38	39	40	41	42	43	44	45		
pipeIdx	6															7
refineIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0
pStateCtx	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62

En una realización preferida el probIdx se define a QP26. Basándose en el valor de inicialización de 8 bits el estado interno (valMps, pipeIdx y refineIdx) de un modelo de probabilidad se procesa tal como se describe en el siguiente pseudocódigo:

```
n = ( probIdx << 8 ) - m * 26
fullCtxState = max( 0, min( 3839, ( m * max( 0,min( 51, SliceQPY ) ) ) ) + n + 128 )
remCtxState = fullCtxState & 255
preCtxState = fullCtxState >> 8
si ( preCtxState  < 8 )  {
     pipeIdx = 7 - preCtxState
     valMPS = 0
} si no  {
     pipeIdx = preCtxState - 8
     valMPS = 1
}
desviación  = { 3, 7, 5, 7, 10, 14, 16, 1 }
si ( pipeIdx = = 0 )  {
      si ( remCtxState <= 127 )
           remCtxState = 127 - remCtxState
      si no
           remCtxState = remCtxState - 128
     refineIdx = ( ( remCtxState << 1 ) * desviación ) >> 8
} si no  {
           si ( valMPS = = 0 )
                remCtxState = 255 - remCtxState
           refineIdx = (remCtxState * desviación [pipeIdx]) >> 8
      }
```

Tal como se muestra en el pseudocódigo el refineIdx se calcula mediante interpolación lineal entre el intervalo del pipeIdx y cuantificando el resultado al refineIdx correspondiente. La desviación especifica el número total de refineIdx para cada pipeIdx. El intervalo [7, 8) de fullCtxState/256 se divide a la mitad. El intervalo [7, 7,5) se mapea en pipeIdx = 0 y valMps = 0 y el intervalo [7,5, 8) se mapea en pipeIdx = 0 y valMps = 1. La figura 16 representa el proceso de derivar el estado interno y visualiza el mapeo de fullCtxState/256 en pStateCtx.

Obsérvese que la pendiente indica la dependencia del probIdx y el QP. Si el slopeIdx del valor de inicialización de 8 bits es igual a 7, el estado interno resultante del modelo de probabilidad es el mismo para todos los QP de segmento, por tanto, el proceso de inicialización del estado interno es independiente del QP actual del segmento.

Es decir, el selector 402 puede inicializar los índices de PIPE que van a usarse en la decodificación de la siguiente porción del flujo de datos tal como el flujo completo o el siguiente segmento, usando el elemento de sintaxis que indica el tamaño de etapa de cuantificación QP usado con el fin de cuantificar los datos de esta porción, tal como los niveles de coeficiente de transformación contenidos en el mismo usando este elemento de sintaxis como índice en una tabla que puede ser común para ambos modos, LC y HE. La tabla, tal como la tabla D, puede comprender índices de PIPE para cada tipo de símbolo, para una referencia respectiva QPref, u otros datos para cada tipo de símbolo. Dependiendo del QP real de la porción actual, el selector puede calcular un valor de índice de PIPE usando la entrada de tabla respectiva a indexada por el QP real y el propio QP, tal como mediante multiplicación de a por (QP-QPref). La única diferencia en el modo de LC y de HE: el selector calcula el resultado simplemente a una precisión menor en el caso de LC en comparación con el modo de HE. El selector puede usar simplemente, por ejemplo, la parte de número entero del resultado de cálculo. En el modo de HE, el resto de mayor precisión, tal como la parte fraccionaria, se usa para seleccionar uno de los índices de refinamiento disponibles para el índice de PIPE respectivo tal como se indica mediante la parte de número entero o menor precisión. El índice de refinamiento se usa en el modo de HE (posiblemente con menor frecuencia también en el modo de LC) con el fin de realizar la adaptación de probabilidad tal como usando el recorrido de tabla anteriormente mencionado. Cuando se dejan los índices disponibles para el índice de PIPE actual al límite superior, entonces se selecciona el índice de PIPE superior junto con minimizar el índice de refinamiento. Cuando se dejan los índices disponibles para el índice de PIPE actual al límite inferior, entonces se selecciona el siguiente índice de PIPE inferior junto con maximizar el índice de refinamiento al máximo disponible para el nuevo índice de PIPE. El índice de PIPE junto con el índice de

refinamiento define el estado de probabilidad, pero para la selección entre los flujos parciales, el selector simplemente usa el índice de PIPE. El índice de refinamiento simplemente sirve para rastrear la probabilidad más estrechamente, o con una precisión más fina.

Sin embargo, la discusión anterior también mostró que puede lograrse un ajuste a escala de la complejidad de manera independiente del concepto de codificación de PIPE de las figuras 7 - 10 o CABAC, usando un decodificador tal como se muestra en la figura 12. El decodificador de la figura 12 es para decodificar un flujo 601 de datos en el que se codifican datos de medios, y comprende un conmutador 600 de modo configurado para activar un modo de baja complejidad o un modo de alta eficiencia dependiendo del flujo 601 de datos, así como un desimbolizador 602 configurado para desimbolizar una secuencia 603 de símbolos obtenida (o bien directamente o bien mediante decodificación entrópica, por ejemplo) a partir del flujo 601 de datos para obtener elementos 604 de sintaxis con valor de número entero usando una función de mapeo controlable mediante un parámetro de control, para mapear un dominio de palabras de secuencia de símbolos en un codominio de los elementos de sintaxis con valor de número entero. Un reconstructor 605 está configurado para reconstruir los datos 606 de medios basándose en los elementos de sintaxis con valor de número entero. El desimbolizador 602 está configurado para realizar la desimbolización de tal manera que el parámetro de control varía según el flujo de datos a una primera tasa en el caso de que esté activado el modo de alta eficiencia y el parámetro de control es constante independientemente del flujo de datos o cambia dependiendo del flujo de datos, pero a una segunda tasa inferior a la primera tasa en el caso de que esté activado el modo de baja complejidad, tal como se ilustra mediante la flecha 607. Por ejemplo, el parámetro de control puede variar según símbolos anteriormente desimbolizados.

Algunas de las realizaciones anteriores emplearon el aspecto de la figura 12. Los elementos de sintaxis *coeff_abs_minus3* y MVD dentro de la secuencia 327 se binarizaron, por ejemplo, en el desimbolizador 314 dependiendo del modo seleccionado tal como se indica mediante 407, y el reconstructor 605 usó estos elementos de sintaxis para la reconstrucción. Evidentemente, ambos aspectos de las figuras 11 y 19 son fácilmente combinables, pero el aspecto de la figura 12 también puede combinarse con otros entornos de codificación.

Véase, por ejemplo, la codificación de diferencia de vector de movimiento indicada anteriormente. El desimbolizador 602 puede estar configurado de tal manera que la función de mapeo usa un código unario truncado para realizar el mapeo dentro de un primer intervalo del dominio de elementos de sintaxis con valor de número entero por debajo de un valor de corte y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de una palabra de código de VLC dentro de un segundo intervalo del dominio de elementos de sintaxis con valor de número entero incluyendo el, y por encima del, valor de corte, en el que el decodificador puede comprender un decodificador 608 entrópico configurado para derivar varios primeros elementos binarios del código unario truncado a partir del flujo 601 de datos usando decodificación entrópica con estimación de probabilidad variable y varios segundos elementos binarios de la palabra de código de VLC usando un modo de derivación de equiprobabilidad constante. En el modo de HE, la codificación entrópica puede ser más compleja que la codificación de LC tal como se ilustra mediante la flecha 609. Es decir, puede aplicarse adaptabilidad de contexto y/o adaptación de probabilidad en el modo de HE y suprimirse en el modo de LC, o puede ajustarse a escala la complejidad de otras maneras, tal como se expuso anteriormente con respecto a las diversas realizaciones.

En la figura 13 se muestra un codificador adaptado al decodificador de la figura 11, para codificar datos de medios para dar un flujo de datos. Puede comprender un insertador 500 configurado para indicar dentro del flujo 501 de datos una activación de un modo de baja complejidad o un modo de alta eficiencia, un constructor 504 configurado para precodificar los datos 505 de medios para dar una secuencia 506 de elementos de sintaxis, un simbolizador 507 configurado para simbolizar la secuencia 506 de elementos de sintaxis para dar una secuencia 508 de símbolos, una pluralidad de codificadores 310 entrópicos cada uno de los cuales está configurado para convertir secuencias parciales de símbolos en palabras de código del flujo de datos, y un selector 502 configurado para reenviar cada símbolo de la secuencia 508 de símbolos a uno seleccionado de la pluralidad de codificadores 310 entrópicos, en el que el selector 502 está configurado para realizar la selección dependiendo del activado del modo de baja complejidad y el modo de alta eficiencia tal como se ilustra mediante la flecha 511. Opcionalmente puede proporcionarse un entrelazador 510 para entrelazar las palabras de código de los codificadores 310.

En la figura 14 se muestra un codificador adaptado al decodificador de la figura 12, para codificar datos de medios para dar un flujo de datos, como que comprende un insertador 700 configurado para indicar dentro del flujo 701 de datos una activación de un modo de baja complejidad o un modo de alta eficiencia, un constructor 704 configurado para precodificar los datos 705 de medios para dar una secuencia 706 de elementos de sintaxis que comprende un elemento de sintaxis con valor de número entero, y un simbolizador 707 configurado para simbolizar el elemento de sintaxis con valor de número entero usando una función de mapeo controlable mediante un parámetro de control, para mapear un dominio de elementos de sintaxis con valor de número entero en un codominio del palabras de secuencia de símbolos, en el que el simbolizador 707 está configurado para realizar la simbolización de tal manera que el parámetro de control varía según el flujo de datos a una primera tasa en el caso de que esté activado el modo de alta eficiencia y el parámetro de control es constante independientemente del flujo de datos o cambia dependiendo del flujo de datos, pero a una segunda tasa inferior a la primera tasa en el caso de que esté activado el

modo de baja complejidad tal como se ilustra mediante la flecha 708. El resultado de simbolización se codifica en el flujo 701 de datos.

De nuevo, debe mencionarse que la realización de la figura 14 puede transferirse fácilmente a la realización de codificación/decodificación aritmética binaria adaptativa de contexto anteriormente mencionada: el selector 509 y los codificadores 310 entrópicos se condensarán para dar un codificador aritmético binario adaptativo de contexto que emitirá el flujo 401 de datos directamente y seleccionará el contexto para un elemento binario que va a derivarse actualmente a partir del flujo de datos. Esto es especialmente cierto para la adaptabilidad de contexto y/o adaptabilidad de probabilidad. Ambas funcionalidades/adaptabilidades pueden desactivarse, o diseñarse de manera más relajada, durante el modo de baja complejidad.

Anteriormente se ha indicado brevemente que la capacidad de conmutación de modo explicada con respecto a algunas de las realizaciones anteriores puede omitirse, según realizaciones alternativas. Para aclarar esto, se hace referencia a la figura 16, que resume la descripción anterior en la medida en que simplemente la eliminación de la capacidad de conmutación de modo distingue la realización de la figura 16 de las realizaciones anteriores. Además, la siguiente descripción revelará las ventajas resultantes de inicializar las estimaciones de probabilidad de los contextos usando parámetros menos precisos para la pendiente y desviación en comparación, por ejemplo, con la norma H.264.

En particular, la figura 16 muestra un decodificador para decodificar un vídeo 405 a partir de un flujo 401 de datos en el que se codifican las componentes horizontal y vertical de diferencias de vector de movimiento usando binarizaciones de las componentes horizontal y vertical, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado. El valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es 2 y el código de Golomb exponencial tiene un orden de 1. El decodificador comprende un decodificador 409 entrópico configurado para derivar, para las componentes horizontal y vertical de las diferencias de vector de movimiento, el código unario truncado a partir del flujo de datos usando decodificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y el código de Golomb exponencial usando un modo de derivación de equiprobabilidad constante para obtener las binarizaciones de las diferencias de vector de movimiento. Más precisamente, tal como se describió anteriormente, el decodificador 409 entrópico puede estar configurado para derivar el número de elementos 326 binarios de las binarizaciones a partir del flujo 401 de datos usando decodificación entrópica binaria tal como el esquema de CABAC anteriormente mencionado, o decodificación de PIPE binaria, es decir usando la construcción que implica varios decodificadores 322 entrópicos que funcionan en paralelo junto con un selector/asignador respectivo. Un desimbolizador 314 desbinariza las binarizaciones de los elementos de sintaxis de diferencia de vector de movimiento para obtener valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento, y un reconstructor 404 reconstruye el vídeo basándose en los valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento.

Con el fin de explicar esto con más detalle, se hace referencia brevemente a la figura 18. 800 muestra de manera representativa una diferencia de vector de movimiento, es decir un vector que representa un residuo de predicción entre un vector de movimiento predicho y un vector de movimiento real/reconstruido. También se ilustran las componentes 802x y 802y horizontal y vertical. Pueden transmitirse en unidades de posiciones de píxeles, es decir paso de píxel, o posiciones de sub-pel tales como la mitad del paso de píxel o una cuarta parte del mismo o similares. Las componentes $802_{x,y}$ horizontal y vertical tienen valores de números enteros. Su dominio alcanza desde cero hasta el infinito. El valor de signo puede gestionarse por separado y no se considera adicionalmente en este caso. Dicho de otro modo, la descripción explicada resumidamente en el presente documento se centra en la magnitud de las diferencias 802x,y de vector de movimiento. El dominio se ilustra en 804. En el lado derecho del eje 804 de dominio, la figura 19 ilustra, asociadas con los valores posibles de la componente 802x,y dispuestos verticalmente unos encima de otros, las binarizaciones en las que se mapea (binariza) el valor posible respectivo. Tal como puede observarse, debajo del valor de corte de 2, simplemente se produce el código 806 unario truncado, mientras que la binarización también tiene, como sufijo, el código de Golomb exponencial de orden 808 de valores posibles iguales a o mayores que el valor de corte de 2 con el fin de continuar la binarización para el resto del valor de número entero por encima del valor de corte menos 1. Para todos los elementos binarios, simplemente se proporcionan dos contextos: uno para la primera posición de elemento binario de binarizaciones de componentes 802x,y horizontal y vertical, y uno adicional para la segunda posición de elemento binario del código 806 unario truncado de ambas componentes 802x,y horizontal y vertical. Para la posición de elemento binario del código 808 de Golomb exponencial, se usa el modo de derivación de equiprobabilidad por el decodificador 409 entrópico. Es decir, se supone que ambos valores de elemento binario se producen con igual probabilidad. La estimación de probabilidad para estos elementos binarios es fija. En comparación con lo mismo, la estimación de probabilidad

asociada con los dos contextos que acaban de mencionarse de los elementos binarios del código 806 unario truncado se adapta de manera continua durante la decodificación.

Antes de describir con más detalle, en cuanto a cómo puede implementarse el decodificador 409 entrópico, según la descripción anterior, con el fin de realizar las tareas que acaban de mencionarse, la descripción se centra ahora en una posible implementación del reconstructor 404 que usa las diferencias 800 de vector de movimiento y los valores de número entero de las mismas tal como se obtienen mediante el desimbolizador 314 volviendo a binarizar los elementos binarios de los códigos 106 y 108, ilustrándose la nueva binarización en la figura 18 usando la flecha 810. En particular, el reconstructor 404 puede recuperar, tal como se describió anteriormente, a partir del flujo 401 de datos información referente a una subdivisión de una imagen actualmente reconstruida en bloques entre los cuales al menos algunos se someten a predicción con compensación de movimiento. La figura 19 muestra una imagen que va a reconstruirse de manera representativa en 820 y bloques de la subdivisión que acaba de mencionarse de la imagen 120 para los que se usa predicción con compensación de movimiento para predecir el contenido de imagen en la misma en 822. Tal como se describió con respecto a las figuras 2A-2C, hay diferentes posibilidades para la subdivisión y los tamaños de los bloques 122. Con el fin de evitar una transmisión para una diferencia 800 de vector de movimiento para cada uno de estos bloques 122, el reconstructor 404 puede aprovechar un concepto de fusión según el cual el flujo de datos transmite adicionalmente información de fusión además de la información de subdivisión o, en ausencia de información de subdivisión, además del hecho de que la subdivisión es fija. La información de fusión indica al reconstructor 404 cuáles de los bloques 822 forman un grupo de fusión. Mediante esta medida, el reconstructor 404 puede aplicar una determinada diferencia 800 de vector de movimiento a un grupo de fusión completo de los bloques 822. Naturalmente, en el lado de codificación, la transmisión de la información de fusión se somete a un compromiso entre coste de transmisión de subdivisión (si está presente), el coste de transmisión de información de fusión y el coste de transmisión de diferencia de vector de movimiento que disminuye al aumentar el tamaño de los grupos de fusión. Por otro lado, aumentar el número de bloques por grupo de fusión reduce la adaptación de la diferencia de vector de movimiento para este grupo de fusión a las necesidades reales de los bloques individuales del grupo de fusión respectivo, proporcionando así predicciones con compensación de movimiento menos precisas de las diferencias de vector de movimiento de estos bloques y necesitando un mayor coste de transmisión para transmitir el residuo de predicción en forma, por ejemplo, de nivel de coeficiente de transformación. Por consiguiente, se encuentra un compromiso en el lado de codificación de una manera apropiada. Sin embargo, en cualquier caso, el concepto de fusión da como resultado que las diferencias de vector de movimiento para los grupos de fusión muestran menos intercorrelación espacial. Véase, por ejemplo, la figura 19 que ilustra mediante sombreado una pertenencia a un determinado grupo de fusión. Evidentemente, el movimiento real del contenido de imagen en estos bloques ha sido tan similar que el lado de codificación decidió fusionar los bloques respectivos. Sin embargo, la correlación con el movimiento del contenido de imagen en otros grupos de fusión es baja. Por consiguiente, la restricción a usar simplemente un contexto por elemento binario del código 806 unario truncado no tiene un impacto negativo en la eficiencia de codificación entrópica ya que el concepto de fusión ya se adapta de manera suficiente a la intercorrelación espacial entre contenido de imagen vecina. El contexto puede seleccionarse simplemente basándose en el hecho de que el elemento binario forma parte de la binarización de una componente $802_{x,y}$ de diferencia de vector de movimiento y que la posición de elemento binario es o bien 1 o bien 2 debido a que el valor de corte es dos. Por consiguiente, otros elementos binarios / elementos de sintaxis / componentes $802_{x,y}$ de mvd ya decodificados no influyen en la selección de contexto.

Asimismo, el reconstructor 404 puede estar configurado para reducir adicionalmente el contenido de información que va a transferirse mediante las diferencias de vector de movimiento (más allá de la predicción espacial y/o temporal de vectores de movimiento) usando un concepto de predicción de múltiples hipótesis según el cual, en primer lugar, se genera una lista de factores de predicción de vector de movimiento para cada bloque o grupo de fusión, transmitiendo entonces, de manera explícita o implícita, dentro del flujo de datos, información sobre el índice del factor de predicción que va a usarse realmente para predecir la diferencia de vector de movimiento. Véase, por ejemplo, el bloque 122 no sombreado en la figura 20. El reconstructor 404 puede proporcionar diferentes factores de predicción para el vector de movimiento de este bloque tales como prediciendo el vector de movimiento espacialmente tal como desde la izquierda, desde arriba, una combinación de ambos, y así sucesivamente, y prediciendo temporalmente el vector de movimiento a partir del vector de movimiento de una porción ubicada conjuntamente de una imagen anteriormente decodificada del vídeo y combinaciones adicionales de los factores de predicción anteriormente mencionados. Estos factores de predicción se clasifican mediante el reconstructor 404 de una manera predecible que puede estimarse en el lado de codificación. Para ello se transmite algo de información dentro del flujo de datos y se usa por el reconstructor. Es decir, hay algún indicio contenido en el flujo de datos, en cuanto a qué factor de predicción de esta lista ordenada de factores de predicción deberá usarse realmente como factor de predicción para el vector de movimiento de este bloque. Este índice puede transmitirse dentro del flujo de datos para este bloque de manera explícita. Sin embargo, también es posible que en primer lugar se prediga el índice y después simplemente se transmita una predicción del mismo. También existen otras posibilidades. En cualquier caso, el esquema de predicción que acaba de mencionarse permite una predicción muy precisa del vector de movimiento del bloque actual y, por consiguiente, se reduce el requisito de contenido de información impuesto en la diferencia de vector de movimiento. Por consiguiente, la restricción de la codificación entrópica adaptativa de contexto simplemente en dos elementos binarios del código unario truncado y una reducción del valor de corte hasta

2 tal como se describió con respecto a la figura 18, así como la selección del orden del código de Golomb exponencial para que sea 1, no afectan de manera negativa a la eficiencia de codificación dado que las diferencias de vector de movimiento muestran, debido a la alta eficiencia de predicción, un histograma de frecuencia según el cual los valores superiores de las componentes 802x,y de diferencia de vector de movimiento se visitan con menor frecuencia. Incluso la omisión de cualquier distinción entre las componentes horizontal y vertical se ajusta a la predicción eficiente, ya que la predicción tiende a funcionar igual de bien en ambos sentidos de la precisión de predicción que es alta.

Resulta esencial observar que, en la descripción anterior, todos los detalles proporcionados con las figuras 1-15 también pueden transferirse a las entidades mostradas en la figura 16 tales como, por ejemplo, en lo que se refiere a la funcionalidad del desimbolizador 314, el reconstructor 404 y el decodificador 409 entrópico. No obstante, por motivos de completitud, a continuación se explican resumidamente de nuevo algunos de estos detalles.

Para una mejor comprensión del esquema de predicción que acaba de explicarse resumidamente, véase la figura 20. Tal como acaba de describirse, el constructor 404 puede obtener diferentes factores de predicción para un bloque 822 actual o un grupo de fusión actual de bloques, mostrándose estos factores de predicción mediante vectores 824 de línea continua. Los factores de predicción pueden obtenerse mediante predicción espacial y/o temporal en la que, adicionalmente, pueden usarse operaciones de media aritmética o similares de modo que los factores de predicción individuales pueden haberse obtenido por el reconstructor 404 de una manera tal que los mismos se correlacionan entre sí. De manera independiente del modo en el que se han obtenido los vectores 826, el reconstructor 404 secuencializa o clasifica estos factores 126 de predicción en una lista ordenada. Esto se ilustra mediante los números 1 a 4 en la figura 21. Se prefiere si el proceso de clasificación puede determinarse de manera única de modo que el codificador y el decodificador pueden funcionar de manera sincronizada. Entonces, el índice que acaba de mencionarse puede obtenerse por el reconstructor 404 para el bloque, o grupo de fusión, actual a partir del flujo de datos, de manera explícita o implícita. Por ejemplo, puede haberse seleccionado el segundo factor de predicción “2” y el reconstructor 404 añade la diferencia 800 de vector de movimiento a este factor 126 de predicción seleccionando, proporcionando así el vector 128 de movimiento finalmente reconstruido que después se usa para predecir, mediante predicción con compensación de movimiento, el contenido del bloque/grupo de fusión actual. En el caso del grupo de fusión, será posible que el reconstructor 404 comprenda diferencias de vector de movimiento adicionales proporcionadas para bloques del grupo de fusión, con el fin de refinar adicionalmente el vector 128 de movimiento con respecto a los bloques individuales del grupo de fusión.

Por tanto, procediendo adicionalmente con la descripción de las implementaciones de las entidades mostradas en la figura 16, puede suceder que decodificador 409 entrópico está configurado para derivar el código 806 unario truncado a partir del flujo 401 de datos usando decodificación aritmética binaria o codificación de PIPE binaria. Ambos conceptos se han descrito anteriormente. Además, el decodificador 409 entrópico puede estar configurado para usar diferentes contextos para las dos posiciones de elemento binario del código 806 unario truncado o, alternativamente, incluso el mismo contexto para ambos elementos binarios. El decodificador 409 entrópico puede estar configurado para realizar una actualización de estado de probabilidad. El decodificador 409 entrópico puede realizar esto pasando, para un elemento binario actualmente derivado a partir del código 806 unario truncado, de un estado de probabilidad actual asociado con el contexto seleccionado para el elemento binario actualmente derivado, a un nuevo estado de probabilidad dependiendo del elemento binario actualmente derivado. Véanse anteriormente las tablas Next_State_LPS y Next_State_MPS, la consulta de tabla con respecto a la cual se realiza mediante el decodificador entrópico además de las otras etapas 0 a 5 indicadas anteriormente. En la discusión anterior, el estado de probabilidad actual se ha mencionado mediante pState_current. Se define para el contexto respectivo de interés. El decodificador 409 entrópico puede estar configurado para decodificar mediante aritmética binaria un elemento binario que va a derivarse actualmente a partir del código 806 unario truncado cuantificando un valor de anchura de intervalo de probabilidad actual, es decir R, que representa un intervalo de probabilidad actual para obtener un índice de intervalo de probabilidad, q_index, y realizando una subdivisión de intervalo indexando una entrada de tabla entre entradas de tabla usando el índice de intervalo de probabilidad y un índice de estado de probabilidad, es decir p_state, que, a su vez, depende del estado de probabilidad actual asociado con el contexto seleccionado para el elemento binario que va a derivarse actualmente, para obtener una subdivisión del intervalo de probabilidad actual en dos intervalos parciales. En las realizaciones anteriormente explicadas de manera resumida, estos intervalos parciales estaban asociados con el símbolo más probable y el menos probable. Tal como se describió anteriormente, el decodificador 409 entrópico puede estar configurado para usar una representación de ocho bits para el valor de anchura de intervalo de probabilidad actual R escogiendo, por ejemplo, dos o tres bits más significativos de la representación de ocho bits y cuantificando el valor de anchura de intervalo de probabilidad actual. El decodificador 409 entrópico puede estar configurado además para seleccionar de entre los dos intervalos parciales basándose en un valor de estado de desviación a partir de un interior del intervalo de probabilidad actual, concretamente V, actualizar el valor de anchura de intervalo de probabilidad R y el valor de estado de desviación, y deducir un valor del elemento binario que va a derivarse actualmente, usando el intervalo parcial seleccionado y realizar una renormalización del valor de anchura de intervalo de probabilidad actualizado R y el valor de estado de desviación V incluyendo una continuación de lectura de bits a partir del flujo 401 de datos. El decodificador 409 entrópico puede estar configurado, por ejemplo, para decodificar mediante aritmética binaria un elemento binario a

partir del código de Golomb exponencial reduciendo a la mitad el valor de anchura de intervalo de probabilidad actual para obtener una subdivisión del intervalo de probabilidad actual en dos intervalos parciales. La reducción a la mitad corresponde a una estimación de probabilidad que se fija e igual a 0,5. Puede implementarse mediante un simple desplazamiento de bit. El decodificador entrópico puede estar configurado además para derivar, para cada diferencia de vector de movimiento, el código unario truncado de las componentes horizontal y vertical de la diferencia de vector de movimiento respectiva a partir del flujo 401 de datos, antes del código de Golomb exponencial de las componentes horizontal y vertical de la diferencia de vector de movimiento respectiva. Mediante esta medida, el decodificador 409 entrópico puede aprovechar que un número de elementos binarios mayor forman juntos una serie de elementos binarios para los que la estimación de probabilidad es fija, concretamente 0,5. Esto puede acelerar el procedimiento de decodificación entrópica. Por otro lado, el decodificador 409 entrópico puede preferir mantener el orden entre las diferencias de vector de movimiento derivando en primer lugar las componentes horizontal y vertical de una diferencia de vector de movimiento procediendo entonces simplemente a derivar las componentes horizontal y vertical de la siguiente diferencia de vector de movimiento. Mediante esta medida, los requisitos de memoria impuestos sobre la entidad de decodificación, es decir el decodificador de la figura 16, se reducen ya que el desimbolizador 314 puede proceder con la desbinarización de las diferencias de vector de movimiento inmediatamente sin tener que esperar a una exploración para detectar diferencias de vector de movimiento adicionales. Esto se permite mediante la selección de contexto: dado que simplemente está disponible exactamente un contexto por posición de elemento binario del código 806, no tiene que inspeccionarse ninguna interrelación espacial.

Tal como se describió anteriormente, el reconstructor 404 puede predecir espacial y/o temporalmente las componentes horizontal y vertical de vectores de movimiento para obtener los factores 126 de predicción para las componentes horizontal y vertical del vector de movimiento y reconstruir las componentes horizontal y vertical de los vectores de movimiento refinando los factores 826 de predicción usando las componentes horizontal y vertical de las diferencias de vector de movimiento, tal como simplemente añadiendo la diferencia de vector de movimiento al factor de predicción respectivo.

Además, el reconstructor 404 puede estar configurado para predecir las componentes horizontal y vertical de vectores de movimiento de diferentes maneras para obtener una lista ordenada de factores de predicción para la componente horizontal y vertical de vectores de movimiento, obtener un índice de lista a partir del flujo de datos y reconstruir las componentes horizontal y vertical de vectores de movimiento refinando el factor de predicción al que un factor de predicción de la lista al que apunta el índice de lista usando las componentes horizontal y vertical de las diferencias de vector de movimiento.

Además, tal como ya se ha descrito anteriormente, el reconstructor 404 puede estar configurado para reconstruir el vídeo usando la predicción con compensación de movimiento aplicando la componente 802x,y horizontal y vertical de los vectores de movimiento a una glanuralidad espacial definida por una subdivisión de las imágenes del vídeo en bloques en el que el reconstructor 404 puede usar elementos de sintaxis de fusión presentes en el flujo 401 de datos para agrupar los bloques en grupos de fusión y aplicar los valores de número entero de las componentes 802x,y horizontal y vertical de las diferencias de vector de movimiento obtenidos por el binarizador 314, en unidades de grupos de fusión.

El reconstructor 404 puede derivar la subdivisión de las imágenes del vídeo en bloques a partir de una porción del flujo 401 de datos que excluye los elementos de sintaxis de fusión. El reconstructor 404 también puede adaptar las componentes horizontal y vertical del vector de movimiento predeterminado para todos los bloques de un grupo de fusión asociado, o refinar las mismas mediante las componentes horizontal y vertical de las diferencias de vector de movimiento asociadas con los bloques del grupo de fusión.

Únicamente por motivos de completitud, la figura 17 muestra un codificador que se ajusta al decodificador de la figura 16. El codificador de la figura 17 comprende un constructor 504, un simbolizador 507 y un codificador 513 entrópico. El codificador comprende un constructor 504 configurado para codificar de manera predictiva el vídeo 505 mediante predicción con compensación de movimiento usando vectores de movimiento y codificar de manera predictiva los vectores de movimiento prediciendo los vectores de movimiento y estableciendo valores 506 de número entero de componentes horizontal y vertical de diferencias de vector de movimiento para representar un error de predicción de los vectores de movimiento predichos; un simbolizador 507 configurado para binarizar los valores de número entero para obtener binarizaciones 508 de las componentes horizontal y vertical de las diferencias de vector de movimiento, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno; y un codificador 513 entrópico configurado para codificar, para las componentes horizontal y vertical de las diferencias de vector de movimiento, el código unario truncado en el flujo de datos usando codificación entrópica

binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y el código de Golomb exponencial usando un modo de derivación de equiprobabilidad constante. Detalles de implementación posibles adicionales pueden transferirse directamente a partir de la descripción referente al decodificador de la figura 16 al codificador de la figura 17.

Aunque algunos aspectos se han descrito en el contexto de un aparato, resulta evidente que estos aspectos también representan una descripción del método correspondiente, en el que un bloque o dispositivo corresponde a una etapa de método o una característica de una etapa de método. De manera análoga, aspectos descritos en el contexto de una etapa de método también representan una descripción de un bloque o elemento o característica correspondiente de un aparato correspondiente. Algunas o la totalidad de las etapas de método pueden ejecutarse por (o usando) un aparato de hardware, tal como, por ejemplo, un microprocesador, un ordenador programable o un circuito electrónico. En algunas realizaciones, alguna o algunas de las etapas de método más importantes pueden ejecutarse por un aparato de este tipo.

La señal codificada de la invención puede almacenarse en un medio de almacenamiento digital o puede transmitirse en un medio de transmisión tal como un medio de transmisión inalámbrico o un medio de transmisión por cable tal como Internet.

Dependiendo de determinados requisitos de implementación, pueden implementarse realizaciones de la invención en hardware o en software. La implementación puede realizarse usando un medio de almacenamiento digital, por ejemplo, un disquete, un DVD, un Blue-Ray, un CD, una ROM, una PROM, una EPROM, una EEPROM o una memoria FLASH, que tienen señales de control electrónicamente legibles almacenadas en el mismo, que actúan conjuntamente (o que pueden actuar conjuntamente) con un sistema informático programable de tal manera que se realiza el método respectivo. Por tanto, el medio de almacenamiento digital puede ser legible por ordenador.

Algunas realizaciones según la invención comprenden un soporte de datos que tiene señales de control electrónicamente legibles, que pueden actuar conjuntamente con un sistema informático programable, de tal manera que se realiza uno de los métodos descritos en el presente documento.

Generalmente, pueden implementarse realizaciones de la presente invención como producto de programa informático con un código de programa, siendo el código de programa operativo para realizar uno de los métodos cuando se ejecuta el producto de programa informático en un ordenador. El código de programa puede almacenarse, por ejemplo, en un soporte legible por máquina.

Otras realizaciones comprenden el programa informático para realizar uno de los métodos descritos en el presente documento, almacenado en un soporte legible por máquina.

Dicho de otro modo, una realización del método de la invención es, por tanto, un programa informático que tiene un código de programa para realizara uno de los métodos descritos en el presente documento, cuando se ejecuta el programa informático en un ordenador.

Una realización adicional de los métodos de la invención es, por tanto, un soporte de datos (o un medio de almacenamiento digital, o un medio legible por ordenador) que comprende, registrado en el mismo, el programa informático para realizar uno de los métodos descritos en el presente documento. El soporte de datos, el medio de almacenamiento digital o el medio registrado son normalmente tangibles y/o no transitorios.

Una realización adicional del método de la invención es, por tanto, un flujo de datos o una secuencia de señales que representa el programa informático para realizar uno de los métodos descritos en el presente documento. El flujo de datos o la secuencia de señales puede estar configurado, por ejemplo, para transferirse a través de una conexión de comunicación de datos, por ejemplo, a través de Internet.

Una realización adicional comprende medios de procesamiento, por ejemplo, un ordenador, o un dispositivo lógico programable, configurados o adaptados para realizar uno de los métodos descritos en el presente documento.

Una realización adicional comprende un ordenador que tiene instalado en el mismo el programa informático para realizar uno de los métodos descritos en el presente documento.

Una realización adicional según la invención comprende un aparato o un sistema configurado para transferir (por ejemplo, de manera electrónica u óptica) un programa informático para realizar uno de los métodos descritos en el presente documento a un receptor. El receptor puede ser, por ejemplo, un ordenador, un dispositivo móvil, un dispositivo de memoria o similar. El aparato o sistema puede comprender, por ejemplo, un servidor de archivos para transferir el programa informático al receptor.

En algunas realizaciones, puede usarse un dispositivo lógico programable (por ejemplo, una matriz de compuertas programable en el campo) para realizar algunas o todas de las funcionalidades de los métodos descritos en el presente documento. En algunas realizaciones, una matriz de compuertas programable en el campo puede actuar conjuntamente con un microprocesador con el fin de realizar uno de los métodos descritos en el presente documento. Generalmente, los métodos se realizan preferiblemente mediante cualquier aparato de hardware.

Las realizaciones descritas anteriormente son simplemente ilustrativas de los principios de la presente invención. Se entiende que modificaciones y variaciones de las disposiciones y los detalles descritos en el presente documento resultarán evidentes para otros expertos en la técnica. Por tanto, la intención es limitarse únicamente por el alcance de las reivindicaciones de patente a continuación y no por los detalles específicos presentados a modo de descripción y explicación de las realizaciones en el presente documento.

Por tanto, entre otras cosas, la descripción anterior reveló un decodificador para decodificar un vídeo a partir de un flujo de datos en el que se codifican componentes horizontal y vertical de diferencias de vector de movimiento usando binarizaciones de las componentes horizontal y vertical, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno, que comprende un decodificador entrópico configurado para derivar, para las componentes horizontal y vertical de las diferencias de vector de movimiento, el código unario truncado a partir del flujo de datos usando decodificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y el código de Golomb exponencial usando un modo de derivación de equiprobabilidad constante para obtener las binarizaciones de las diferencias de vector de movimiento; un desimbolizador configurado para desbinarizar las binarizaciones de los elementos de sintaxis de diferencia de vector de movimiento para obtener valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento; un reconstructor configurado para reconstruir el vídeo basándose en los valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento. El decodificador 409 entrópico puede estar configurado para decodificar de manera aritmética binaria un elemento binario que va a derivarse actualmente a partir del código 806 unario truncado cuantificando un valor de anchura de intervalo de probabilidad actual que representa un intervalo de probabilidad actual para obtener un índice de intervalo de probabilidad y realizando una subdivisión de intervalo indexando una entrada de tabla entre entradas de tabla usando el índice de intervalo de probabilidad y un índice de estado de probabilidad dependiendo de un estado de probabilidad actual asociado con el contexto seleccionado para el elemento binario que va a derivarse actualmente, para obtener una subdivisión del intervalo de probabilidad actual en dos intervalos parciales. Con respecto a esto, el decodificador 409 entrópico puede estar configurado para usar una representación de 8 bits para el valor de anchura de intervalo de probabilidad actual y escoger 2 ó 3 bits más significativos de la representación de 8 bits en la cuantificación del valor de anchura de intervalo de probabilidad actual. El decodificador 409 entrópico puede estar configurado para seleccionar de entre los dos intervalos parciales basándose en un valor de estado de desviación a partir de un interior del intervalo de probabilidad actual, actualizar el valor de anchura de intervalo de probabilidad y el valor de estado de desviación, y deducir un valor del elemento binario que va a derivarse actualmente, usando el intervalo parcial seleccionado y realizar una renormalización del valor de anchura de intervalo de probabilidad actualizado y el valor de estado de desviación incluyendo una continuación de lectura de bits a partir del flujo 401 de datos. Y el decodificador 409 entrópico puede estar configurado para decodificar mediante aritmética binaria, en el modo de derivación de equiprobabilidad constante, un elemento binario a partir del código de Golomb exponencial reduciendo a la mitad el valor de anchura de intervalo de probabilidad actual para obtener una subdivisión del intervalo de probabilidad actual en dos intervalos parciales.

Además, la descripción anterior también reveló, entre otras cosas, un codificador para codificar un vídeo para dar un flujo de datos, que comprende un constructor configurado para codificar de manera predictiva el vídeo mediante predicción con compensación de movimiento usando vectores de movimiento y codificando de manera predictiva los vectores de movimiento prediciendo los vectores de movimiento y estableciendo valores de número entero de componentes horizontal y vertical de diferencias de vector de movimiento para representar un error de predicción de los vectores de movimiento predichos; un simbolizador configurado para binarizar los valores de número entero para obtener binarizaciones de las componentes horizontal y vertical de las diferencias de vector de movimiento, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno; y un codificador entrópico configurado para codificar, para las componentes horizontal y vertical de las diferencias de vector de movimiento, el

código unario truncado en el flujo de datos usando codificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y el código de Golomb exponencial usando un modo de derivación de equiprobabilidad constante. El codificador entrópico puede estar configurado para codificar el código unario truncado en el flujo de datos usando codificación aritmética binaria o codificación de PIPE binaria. El codificador entrópico puede estar configurado para usar diferentes contextos para las dos posiciones de elemento binario del código unario truncado. Además, el codificador entrópico puede estar configurado para realizar una actualización de estado de probabilidad pasando, para un elemento binario actualmente codificado a partir del código unario truncado, de un estado de probabilidad actual asociado con el contexto seleccionado para el elemento binario actualmente codificado, a un nuevo estado de probabilidad dependiendo del elemento binario actualmente derivado. Adicionalmente, el codificador entrópico puede estar configurado para codificar mediante aritmética binaria un elemento binario que va a codificarse actualmente a partir del código unario truncado cuantificando un valor de anchura de intervalo de probabilidad actual que representa un intervalo de probabilidad actual para obtener un índice de intervalo de probabilidad y realizando una subdivisión de intervalo indexando una entrada de tabla entre entradas de tabla usando el índice de intervalo de probabilidad y un índice de estado de probabilidad dependiendo de un estado de probabilidad actual asociado con el contexto seleccionado para el elemento binario que va a derivarse actualmente, para obtener una subdivisión del intervalo de probabilidad actual en dos intervalos parciales. Con respecto a esto, el codificador entrópico puede estar configurado para usar una representación de 8 bits para el valor de anchura de intervalo de probabilidad actual y escoger 2 ó 3 bits más significativos de la representación de 8 bits en la cuantificación del valor de anchura de intervalo de probabilidad actual. Además, el codificador entrópico puede estar configurado para seleccionar de entre los dos intervalos parciales basándose en el valor de número entero del elemento binario que va a codificarse actualmente, actualizar el valor de anchura de intervalo de probabilidad y una desviación de intervalo de probabilidad usando el intervalo parcial seleccionado y realizar una renormalización del valor de anchura de intervalo de probabilidad y la desviación de intervalo de probabilidad incluyendo una continuación de escritura bits en el flujo de datos. El codificador entrópico puede estar configurado para codificar de manera aritmética binaria un elemento binario a partir del código de Golomb exponencial reduciendo a la mitad el valor de anchura de intervalo de probabilidad actual para obtener una subdivisión del intervalo de probabilidad actual en dos intervalos parciales. El codificador entrópico también puede estar configurado para codificar, para cada diferencia de vector de movimiento, el código unario truncado de las componentes horizontal y vertical de la diferencia de vector de movimiento respectiva en el flujo de datos, antes del código de Golomb exponencial de las componentes horizontal y vertical de la diferencia de vector de movimiento respectiva. El constructor puede estar configurado para predecir espacial y/o temporalmente las componentes horizontal y vertical de los vectores de movimiento para obtener factores de predicción para las componentes horizontal y vertical de vectores de movimiento y determinar las componentes horizontal y vertical de las diferencias de vector de movimiento para refinar los factores de predicción hacia las componentes horizontal y vertical de vectores de movimiento. El constructor puede estar configurado para predecir las componentes horizontal y vertical de los vectores de movimiento de diferentes maneras para obtener una lista ordenada de factores de predicción para las componentes horizontal y vertical de vectores de movimiento, determinar un índice de lista e insertar información que revela el mismo en el flujo de datos y determinar las componentes horizontal y vertical de las diferencias de vector de movimiento para refinar el factor de predicción al que un factor de predicción de la lista al que apunta el índice de lista hacia las componentes horizontal y vertical de vectores de movimiento. El constructor puede estar configurado para codificar el vídeo usando la predicción con compensación de movimiento aplicando las componentes horizontal y vertical de vectores de movimiento a una glanuralidad espacial definida por una subdivisión de las imágenes del vídeo en bloques, en el que el constructor determina, e inserta en el flujo de datos, elementos de sintaxis de fusión para agrupar los bloques en grupos de fusión y aplicar los valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento sometidos a binarización por el desbinarizador, en unidades de grupos de fusión. Con respecto a esto, el constructor puede estar configurado para codificar la subdivisión de las imágenes del vídeo en bloques en una porción del flujo de datos excluyendo los elementos de sintaxis de fusión. El constructor puede estar configurado para adoptar las componentes horizontal y vertical de un vector de movimiento predeterminado para todos los bloques de un grupo de fusión asociado, o refinar las mismas mediante las componentes horizontal y vertical de las diferencias de vector de movimiento asociadas con los bloques del grupo de fusión.

REIVINDICACIONES

1. Decodificador para decodificar un vídeo a partir de un flujo de datos en el que se codifican componentes horizontal y vertical de diferencias de vector de movimiento usando binarizaciones de las componentes horizontal y vertical, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno, que comprende

un decodificador (409) entrópico configurado para derivar, para las componentes horizontal y vertical de las diferencias de vector de movimiento, el código unario truncado a partir del flujo de datos usando decodificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y el código de Golomb exponencial usando un modo de derivación de equiprobabilidad constante para obtener las binarizaciones de las diferencias de vector de movimiento;

un desimbolizador (314) configurado para desbinarizar las binarizaciones de los elementos de sintaxis de diferencia de vector de movimiento para obtener valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento;

un reconstructor (404) configurado para reconstruir el vídeo basándose en los valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento.

2. Decodificador según la reivindicación 1, en el que el decodificador (409) entrópico está configurado para derivar el código (806) unario truncado a partir del flujo (401) de datos usando decodificación aritmética binaria o decodificación de PIPE binaria.

3. Decodificador según la reivindicación 1 ó 2, en el que el decodificador (409) entrópico está configurado para usar diferentes contextos para las dos posiciones de elemento binario del código 806 unario truncado.

4. Decodificador según cualquiera de las reivindicaciones 1 a 3, en el que el decodificador (409) entrópico está configurado para realizar una actualización de estado de probabilidad pasando, para un elemento binario actualmente derivado del código (806) unario truncado, de un estado de probabilidad actual asociado con el contexto seleccionado para el elemento binario actualmente derivado, a un nuevo estado de probabilidad dependiendo del elemento binario actualmente derivado.

5. Decodificador según cualquiera de las reivindicaciones 1 a 4, en el que el decodificador (409) entrópico está configurado para derivar, para cada diferencia de vector de movimiento, el código unario truncado de las componentes horizontal y vertical de la diferencia de vector de movimiento respectiva a partir del flujo de datos, antes del código de Golomb exponencial de las componentes horizontal y vertical de la diferencia de vector de movimiento respectiva.

6. Decodificador según cualquiera de las reivindicaciones 1 a 5, en el que el reconstructor está configurado para predecir espacial y/o temporalmente las componentes horizontal y vertical de vectores de movimiento para obtener factores de predicción para las componentes horizontal y vertical de los vectores de movimiento y reconstruir las componentes horizontal y vertical de los vectores de movimiento refinando los factores 826 de predicción usando las componentes horizontal y vertical de las diferencias de vector de movimiento.

7. Decodificador según cualquiera de las reivindicaciones 1 a 6, en el que el reconstructor está configurado para predecir las componentes horizontal y vertical de vectores de movimiento de diferentes maneras para obtener una lista ordenada de factores de predicción para las componentes horizontal y vertical de vectores de movimiento, obtener un índice de lista a partir del flujo de datos y reconstruir las componentes horizontal y vertical de vectores de movimiento refinando el factor de predicción al que un factor de predicción de la lista al que apunta el índice de lista usando las componentes horizontal y vertical de las diferencias de vector de movimiento.

8. Decodificador según la reivindicación 7, en el que el reconstructor está configurado para reconstruir el vídeo usando predicción con compensación de movimiento mediante el uso de las componentes horizontal y vertical de vectores de movimiento.

9. Decodificador según la reivindicación 8, en el que el reconstructor está configurado para reconstruir el vídeo usando la predicción con compensación de movimiento aplicando las componentes horizontal y vertical de vectores de movimiento a una glanuralidad espacial definida por una subdivisión de las imágenes del vídeo en bloques, en el que el reconstructor usa elementos de sintaxis de fusión presentes en el flujo de datos para agrupar los bloques en grupos de fusión y aplicar los valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento obtenidos por el desbinarizador, en unidades de grupos de fusión.

10. Decodificador según la reivindicación 9, en el que el reconstructor está configurado para derivar la subdivisión de las imágenes del vídeo en bloques a partir de una porción del flujo de datos excluyendo los elementos de sintaxis de fusión.

11. Decodificador según la reivindicación 9 ó 10, en el que el reconstructor está configurado para adoptar las componentes horizontal y vertical de un vector de movimiento predeterminado para todos los bloques de un grupo de fusión asociado, o refinar las mismas mediante las componentes horizontal y vertical de las diferencias de vector de movimiento asociadas con los bloques del grupo de fusión.

12. Decodificador según cualquiera de las reivindicaciones 1 a 11, en el que el flujo de datos tiene codificado en el mismo un mapa de profundidad.

13. Codificador para codificar un vídeo para dar un flujo de datos, que comprende

un constructor (504) configurado para codificar de manera predictiva el vídeo mediante predicción con compensación de movimiento usando vectores de movimiento y codificando de manera predictiva los vectores de movimiento prediciendo los vectores de movimiento y estableciendo valores de número entero de componentes horizontal y vertical de diferencias de vector de movimiento para representar un error de predicción de los vectores de movimiento predichos;

un simbolizador (507) configurado para binarizar los valores de número entero para obtener binarizaciones de las componentes horizontal y vertical de las diferencias de vector de movimiento, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno; y

un codificador (513) entrópico configurado para codificar, para las componentes horizontal y vertical de las diferencias de vector de movimiento, el código unario truncado en el flujo de datos usando codificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y usando el código de Golomb exponencial un modo de derivación de equiprobabilidad constante.

14. Codificador según la reivindicación 13, en el que el flujo de datos tiene codificado en el mismo un mapa de profundidad.

15. Método para decodificar un vídeo a partir de un flujo de datos en el que se codifican componentes horizontal y vertical de diferencias de vector de movimiento usando binarizaciones de las componentes horizontal y vertical, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno, que comprende

para las componentes horizontal y vertical de las diferencias de vector de movimiento, derivar el código unario truncado a partir del flujo de datos usando decodificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y usando el código de Golomb exponencial un modo de derivación de equiprobabilidad constante para obtener las binarizaciones de las diferencias de vector de movimiento;

desbinarizar las binarizaciones de los elementos de sintaxis de diferencia de vector de movimiento para obtener valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento;

reconstruir el vídeo basándose en los valores de número entero de las componentes horizontal y vertical de las diferencias de vector de movimiento.

16. Método para codificar un vídeo para dar un flujo de datos, que comprende

codificar de manera predictiva el vídeo mediante predicción con compensación de movimiento usando vectores de movimiento y codificar de manera predictiva los vectores de movimiento prediciendo los vectores de movimiento y estableciendo valores de número entero de componentes horizontal y vertical de diferencias de vector de movimiento para representar un error de predicción de los vectores de movimiento predichos;

binarizar los valores de número entero para obtener binarizaciones de las componentes horizontal y vertical de las diferencias de vector de movimiento, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno; y para las componentes horizontal y vertical de las diferencias de vector de movimiento, codificar el código unario truncado en el flujo de datos usando codificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y usando el código de Golomb exponencial un modo de derivación de equiprobabilidad constante.

17. Flujo de datos codificado usando un método para codificar un vídeo para dar un flujo de datos, comprendiendo el método codificar de manera predictiva el vídeo mediante predicción con compensación de movimiento usando vectores de movimiento y codificar de manera predictiva los vectores de movimiento prediciendo los vectores de movimiento y estableciendo valores de número entero de componentes horizontal y vertical de diferencias de vector de movimiento para representar un error de predicción de los vectores de movimiento predichos;

binarizar los valores de número entero para obtener binarizaciones de las componentes horizontal y vertical de las diferencias de vector de movimiento, igualando las binarizaciones un código unario truncado de las componentes horizontal y vertical, respectivamente, dentro de un primer intervalo del dominio de las componentes horizontal y vertical por debajo de un valor de corte, y una combinación de un prefijo en forma del código unario truncado para el valor de corte y un sufijo en forma de un código de Golomb exponencial de las componentes horizontal y vertical, respectivamente, dentro de un segundo intervalo del dominio de las componentes horizontal y vertical incluyendo el, y por encima del, valor de corte, en el que el valor de corte es dos y el código de Golomb exponencial tiene un orden de uno; y para las componentes horizontal y vertical de las diferencias de vector de movimiento, codificar el código unario truncado en el flujo de datos usando codificación entrópica binaria adaptativa de contexto con exactamente un contexto por posición de elemento binario del código unario truncado, que es común para las componentes horizontal y vertical de las diferencias de vector de movimiento, y usando el código de Golomb exponencial un modo de derivación de equiprobabilidad constante.

18. Flujo de datos según la reivindicación 18, en el que el flujo de datos tiene codificado en el mismo un mapa de profundidad.

19. Programa informático que tiene un código de programa para realizar, cuando se ejecuta en un ordenador, un método según cualquiera de las reivindicaciones 15 a 16.

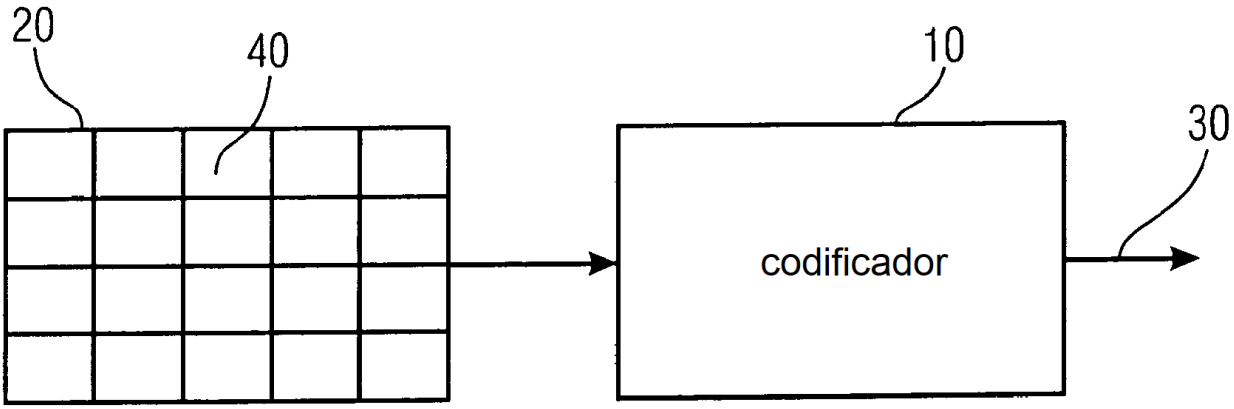

FIGURA 1

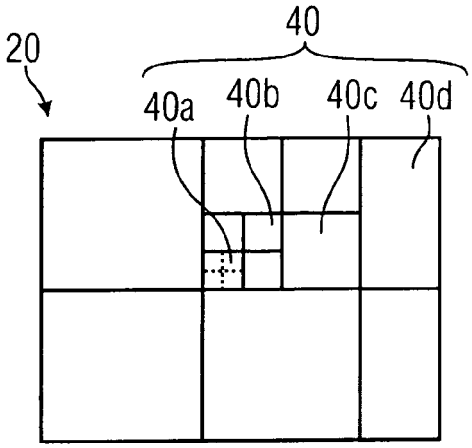

FIGURA 2A

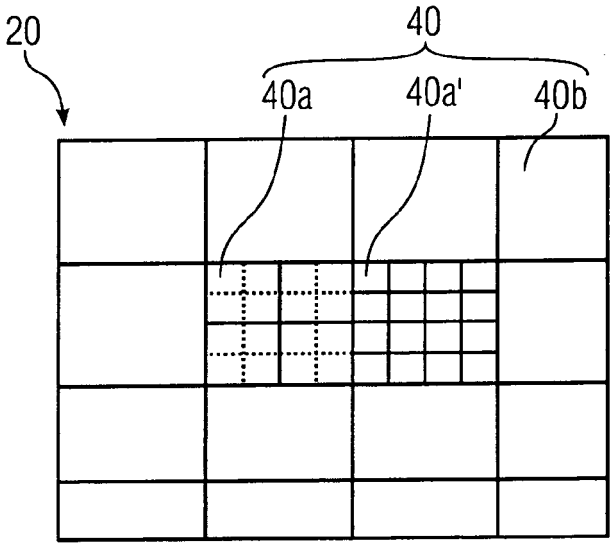

FIGURA 2B

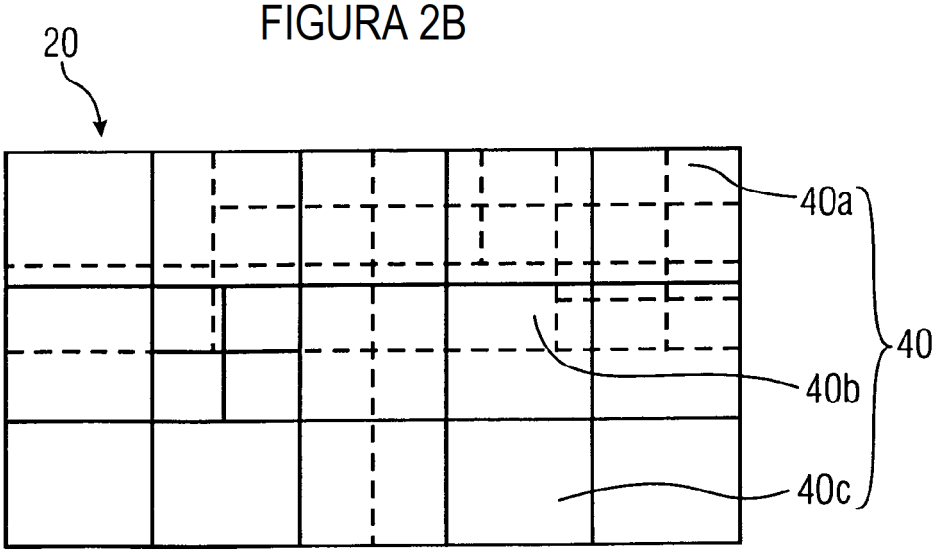

FIGURA 2C

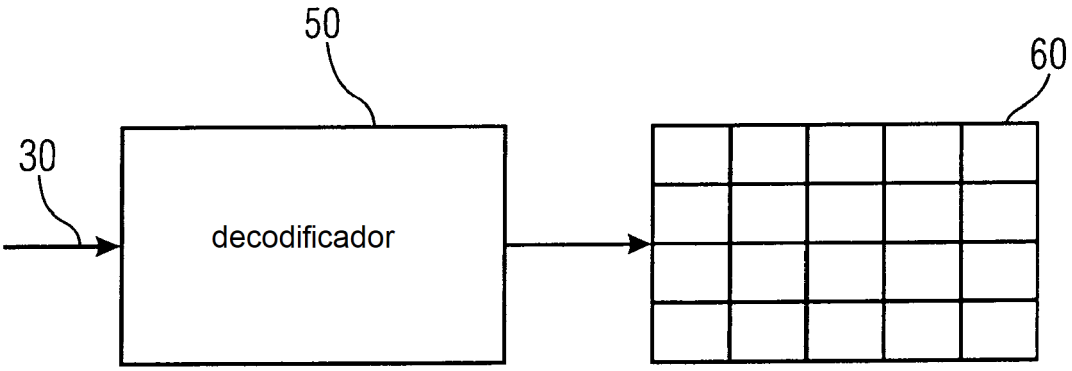

FIGURA 3

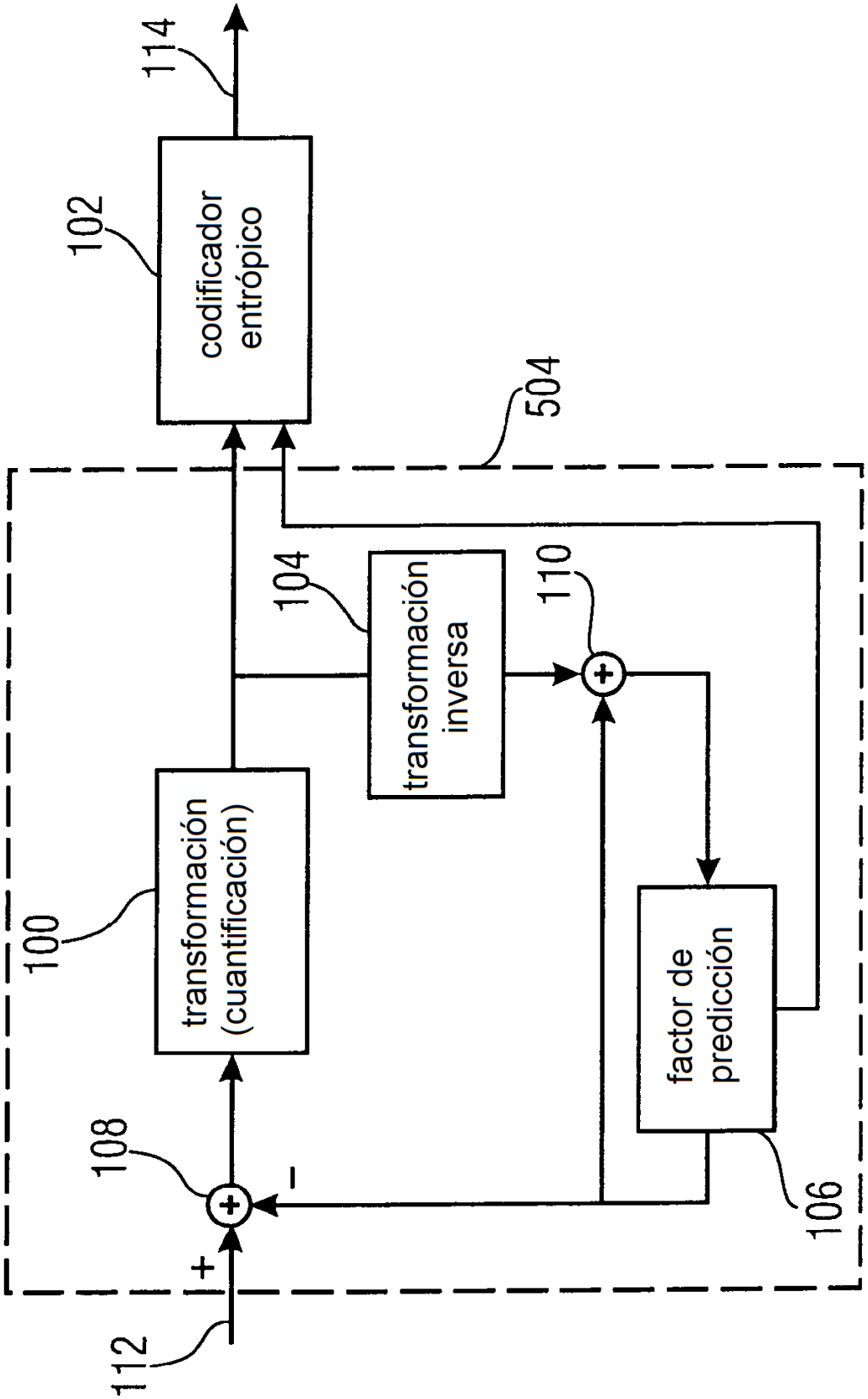
114
102
codificador entrópico
504
104
transformación inversa
110
100
transformación (cuantificación)
factor de predicción
108
106
112
+
-

FIGURA 4

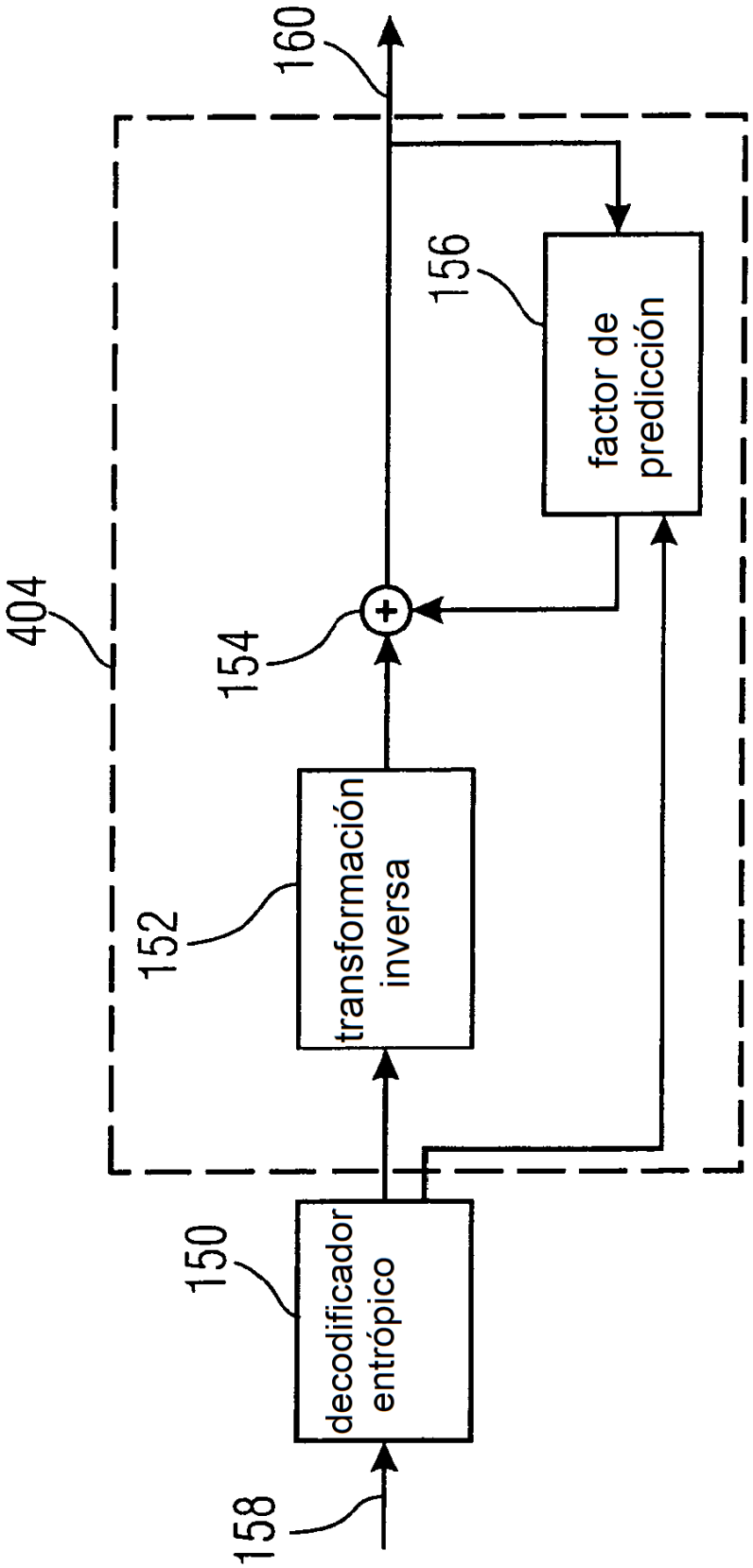
160
156
factor de predicción
404
154
152
transformación inversa
150
decodificador entrópico
158

FIGURA 5

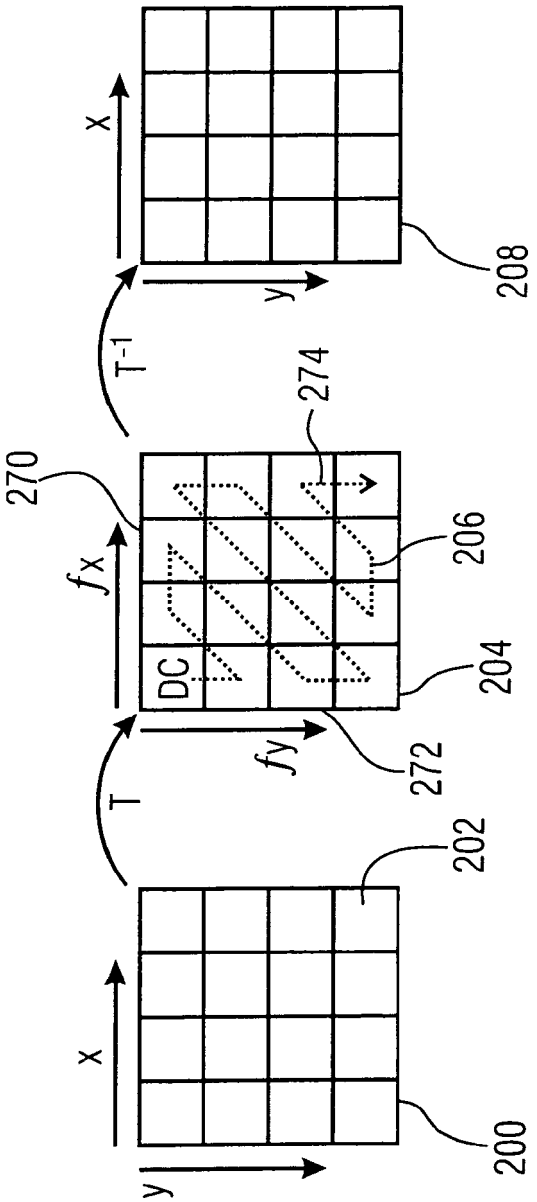

FIGURA 6

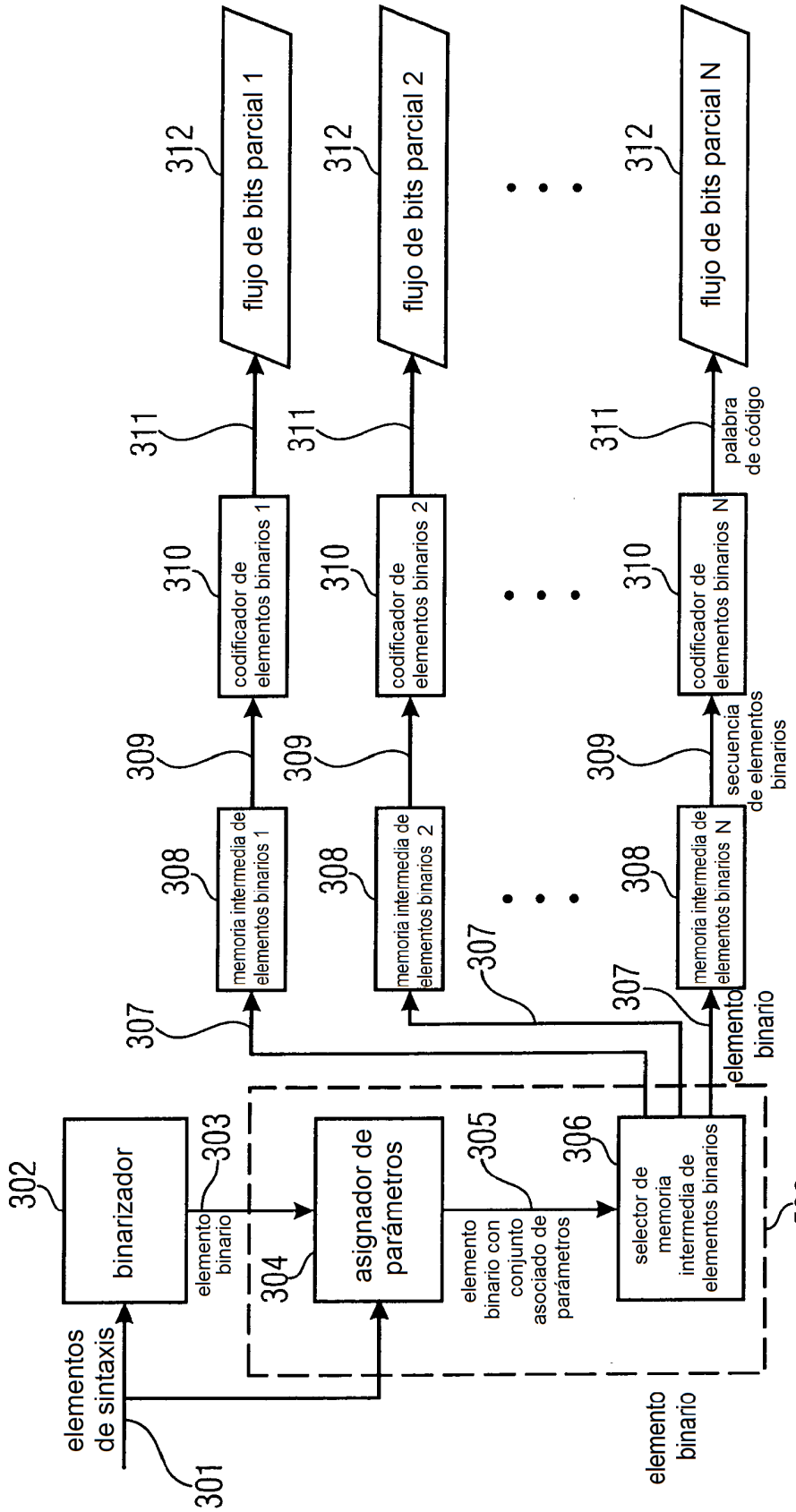
elementos de sintaxis
301
binarizador
302
elemento binario
303
304
asignador de parámetros
elemento binario con conjunto asociado de parámetros
305
306
selector de memoria intermedia de elementos binarios
502
elemento binario
307
memoria intermedia de elementos binarios 1
memoria intermedia de elementos binarios 2
memoria intermedia de elementos binarios N
308
309
secuencia de elementos binarios
codificador de elementos binarios 1
codificador de elementos binarios 2
codificador de elementos binarios N
310
311
palabra de código
flujo de bits parcial 1
flujo de bits parcial 2
flujo de bits parcial N
312

FIGURA 7

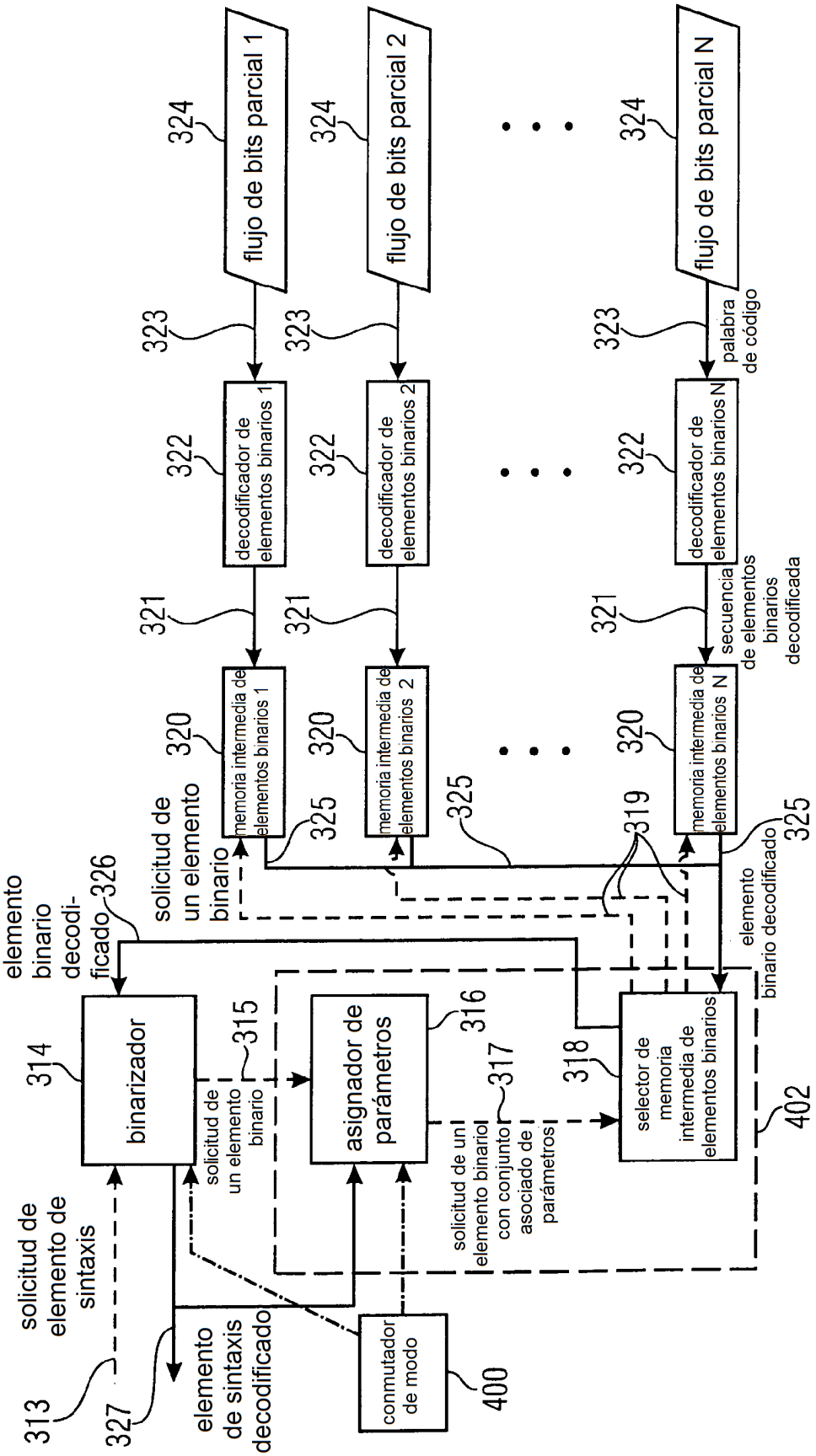
flujo de bits parcial 1
flujo de bits parcial 2
flujo de bits parcial N
palabra de código
decodificador de elementos binarios 1
decodificador de elementos binarios 2
decodificador de elementos binarios N
memoria intermedia de elementos binarios 1
memoria intermedia de elementos binarios 2
memoria intermedia de elementos binarios N
secuencia de elementos binarios decodificada
solicitud de un elemento binario
elemento binario decodificado
elemento binario decodificado
binarizador
asignador de parámetros
selector de memoria intermedia de elementos binarios
conmutador de modo
solicitud de elemento de sintaxis
elemento de sintaxis decodificado
solicitud de un elemento binario
solicitud de un elemento binario con conjunto asociado de parámetros
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
400
402

FIGURA 8

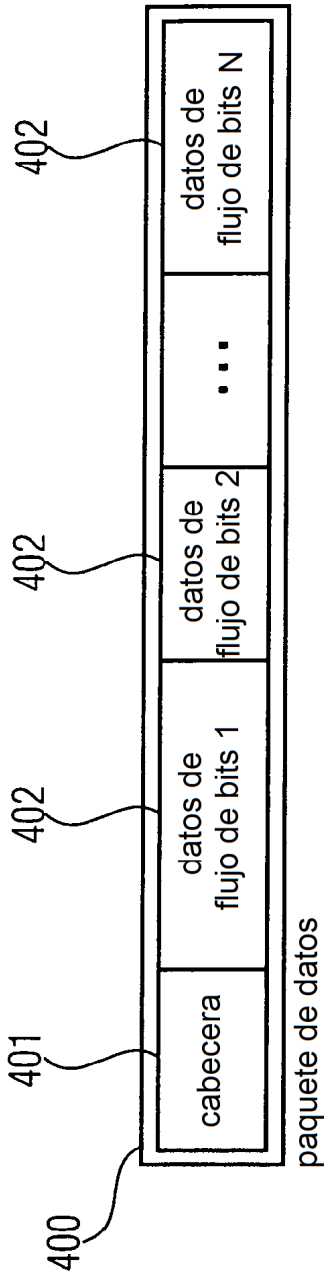

FIGURA 9

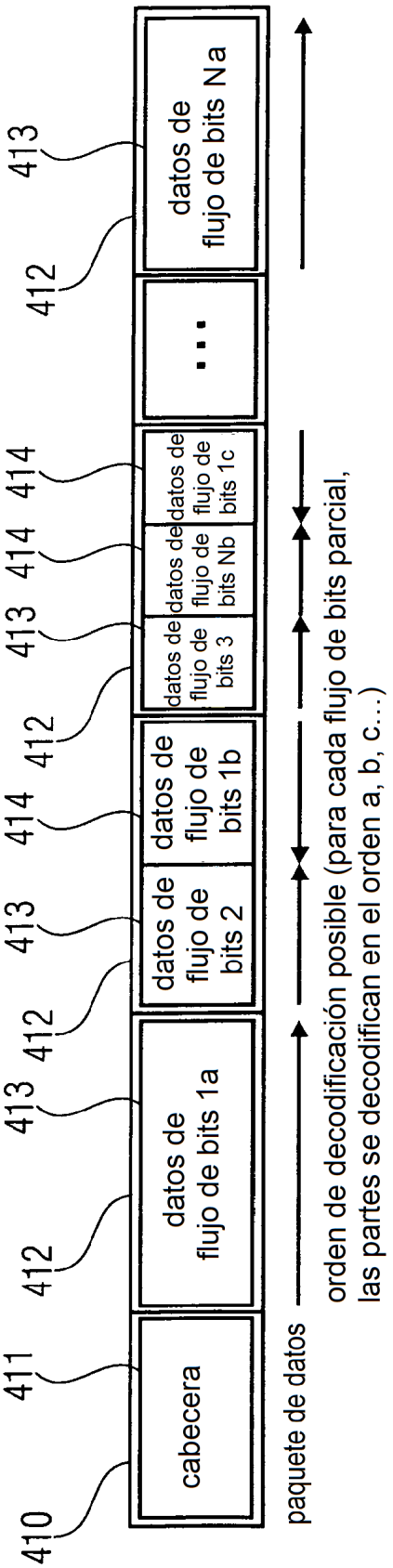

FIGURA 10

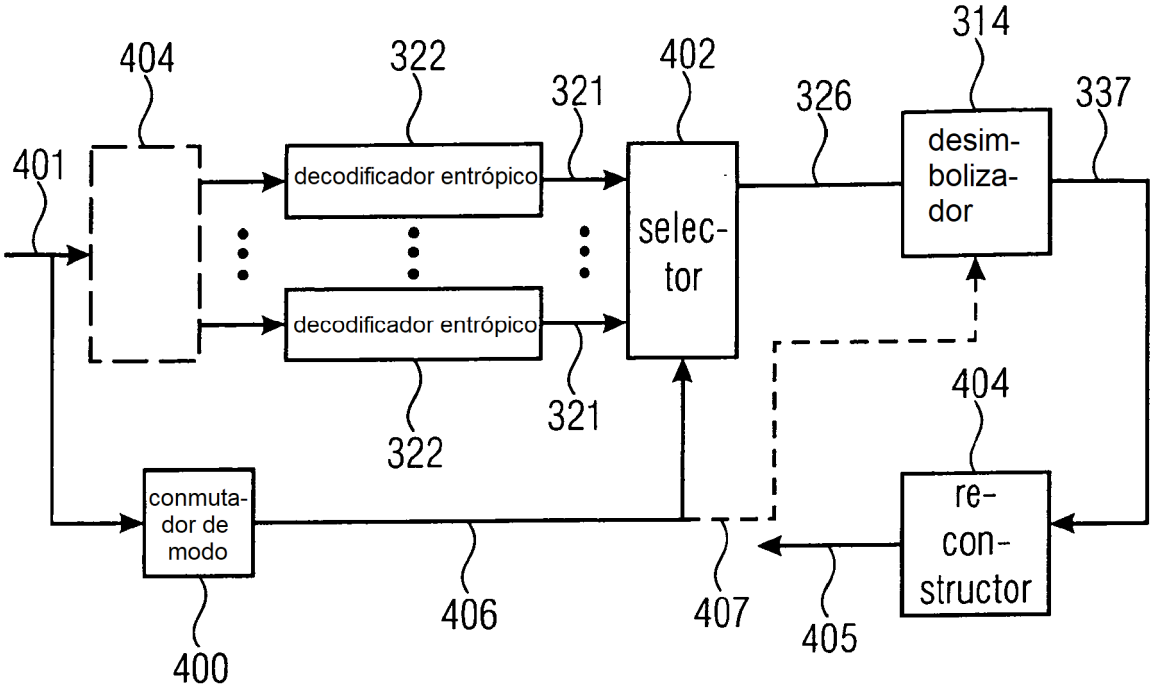

FIGURA 11

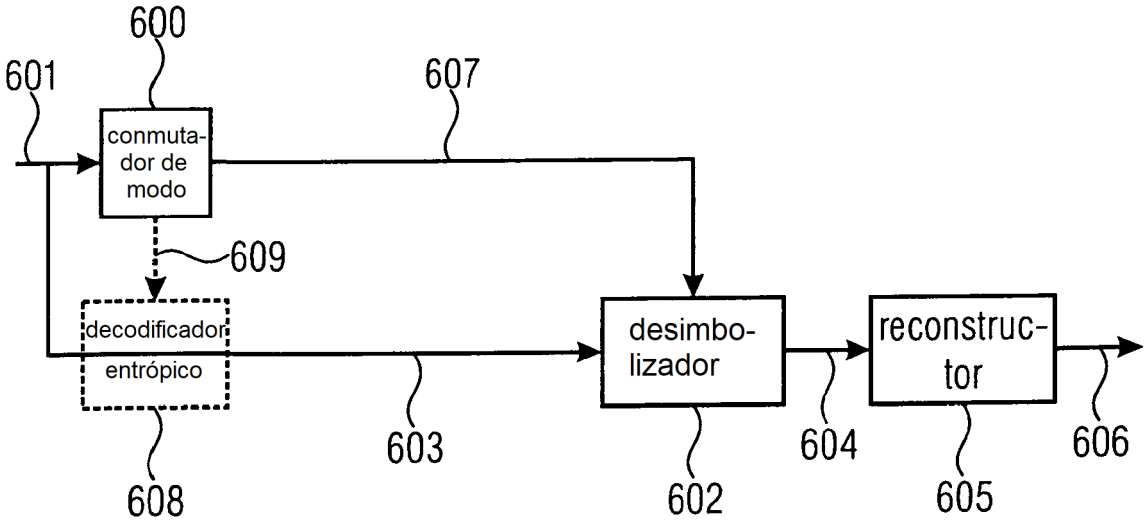

FIGURA 12

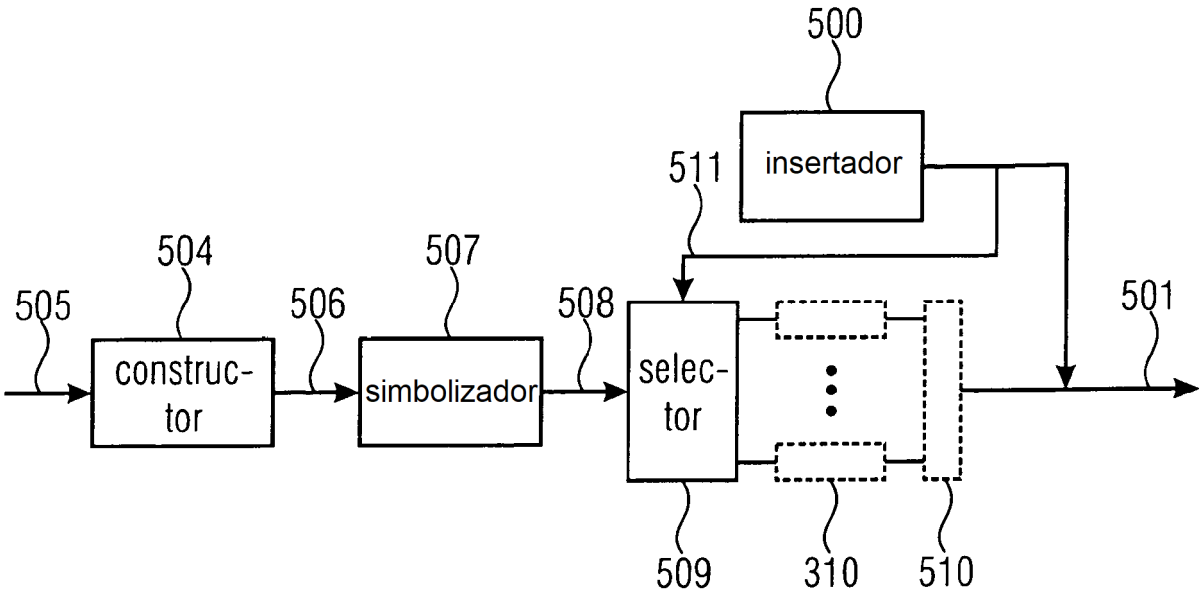

FIGURA 13

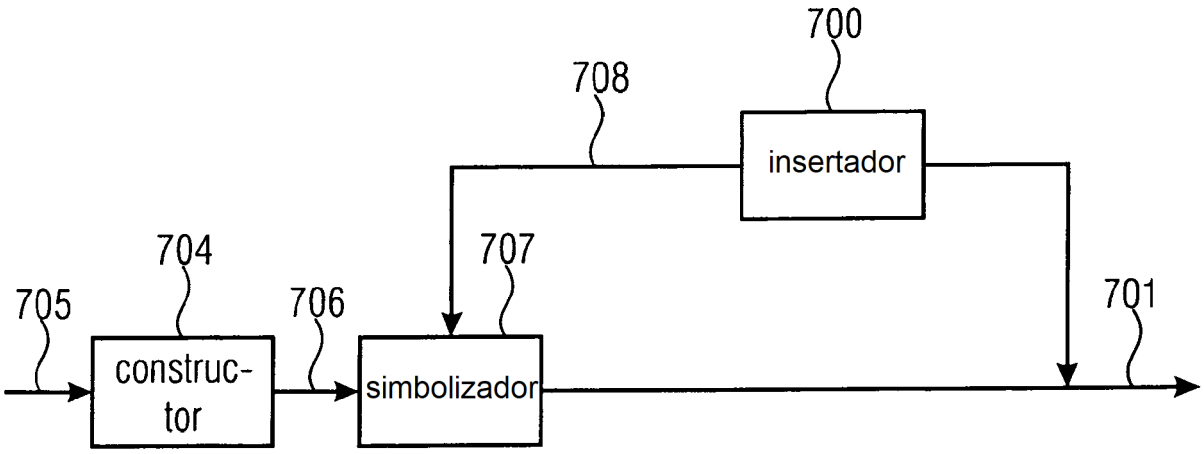

FIGURA 14

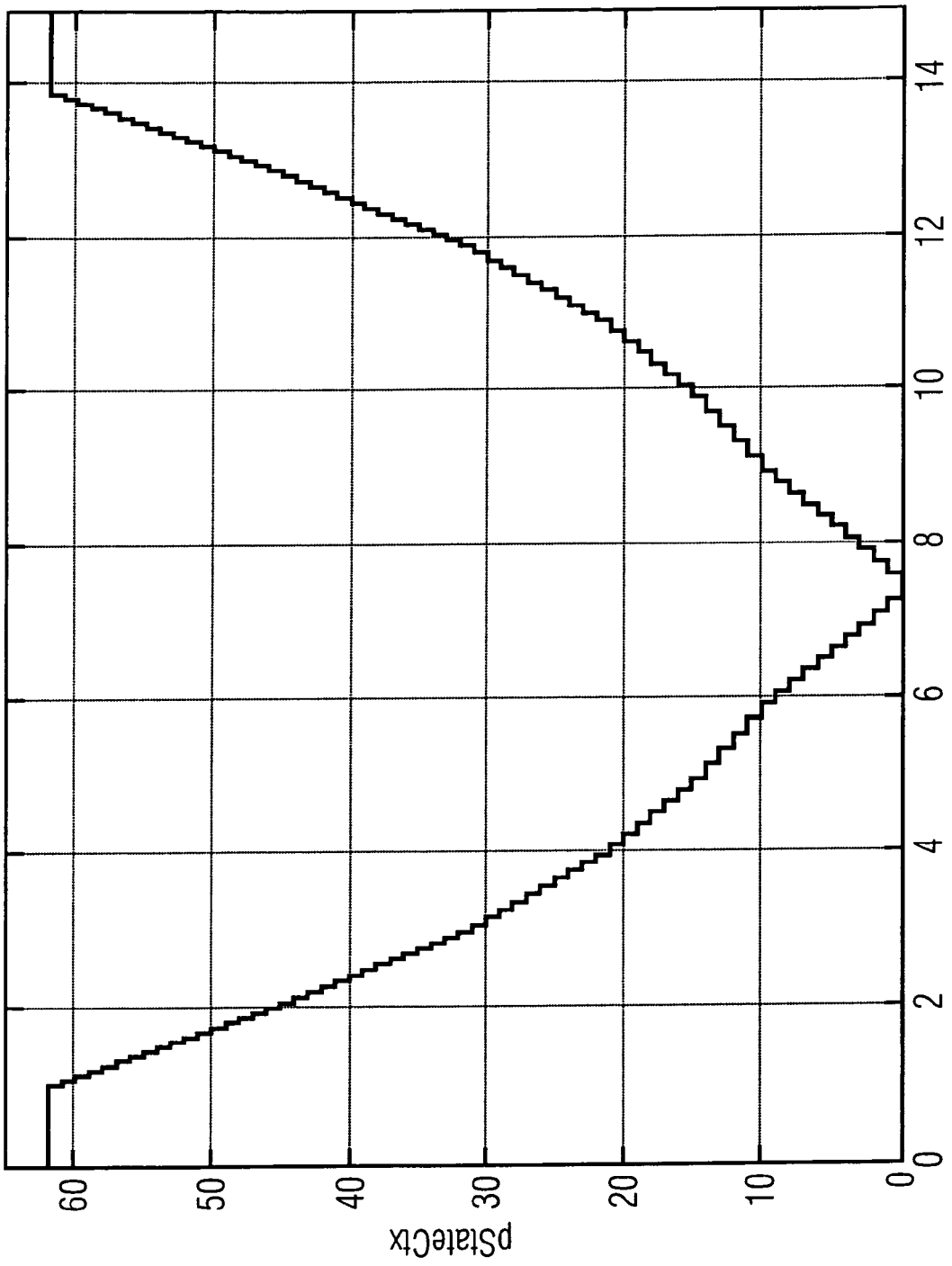

FIGURA 15

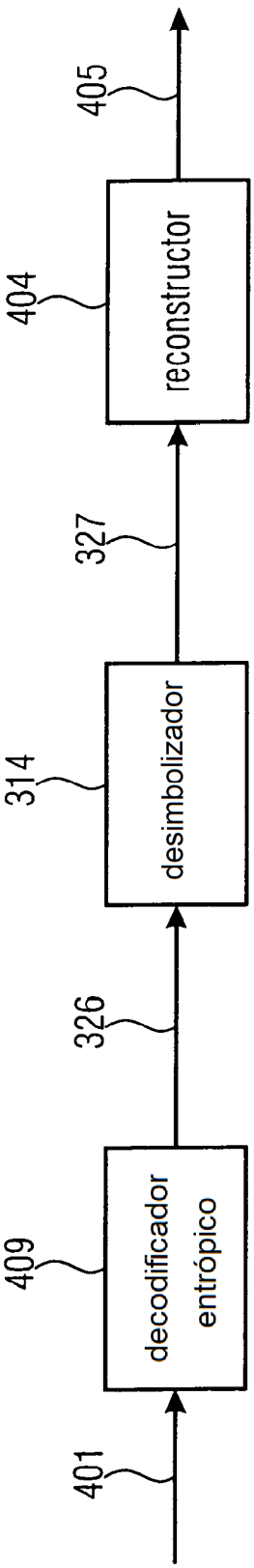

FIGURA 16

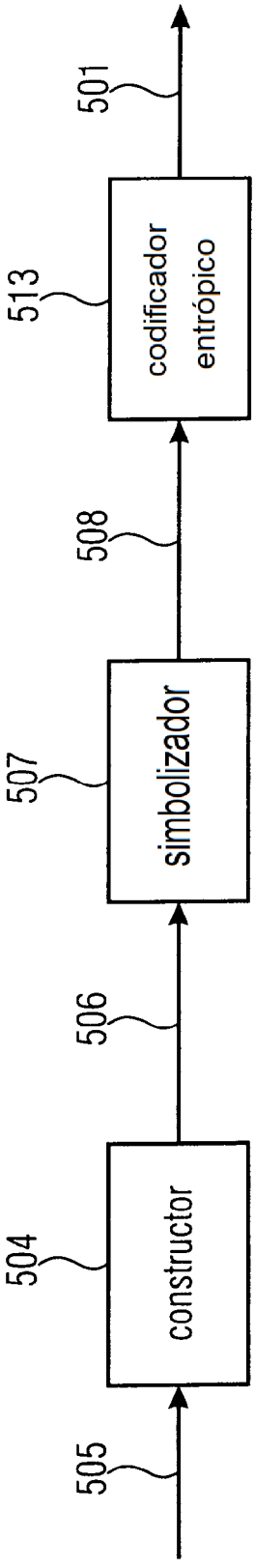

FIGURA 17

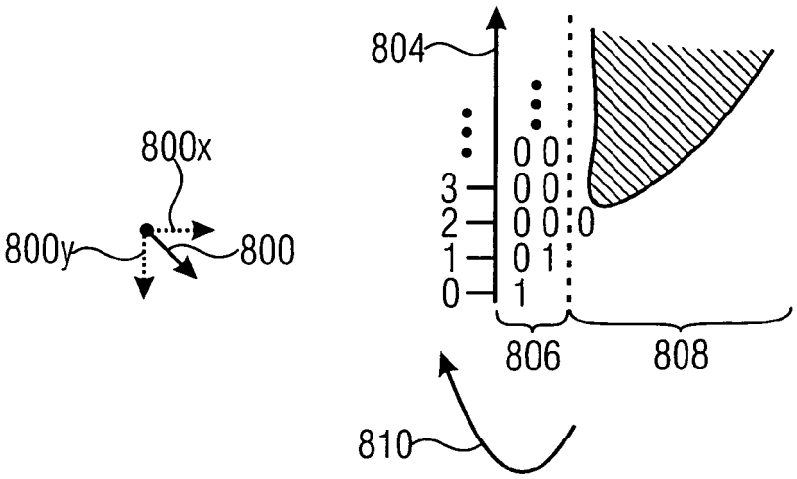

FIGURA 18

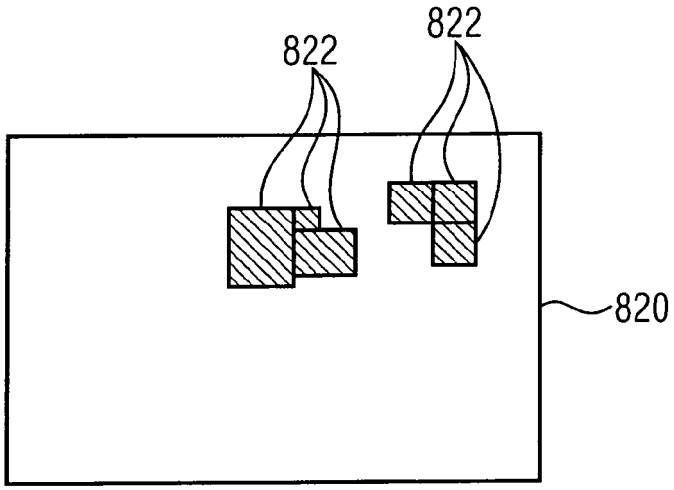

FIGURA 19

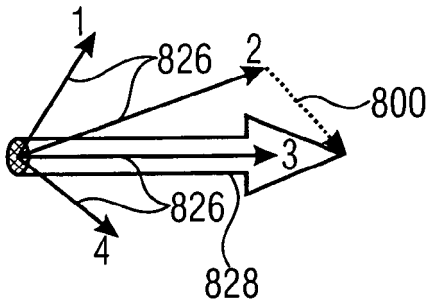

FIGURA 20