(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 750 176**

(51) Int. CI.:

H04N 19/70 (2014.01)
H04N 19/46 (2014.01)
H04N 19/61 (2014.01)
H04N 19/174 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **20.05.2013 PCT/US2013/041764**

(87) Fecha y número de publicación internacional: **19.12.2013 WO13188057**

(96) Fecha de presentación y número de la solicitud europea: **20.05.2013 E 13726965 (0)**

(97) Fecha y número de publicación de la concesión europea: **11.09.2019 EP 2862353**

(54) Título: **Método y aparato para el tratamiento eficaz de cabeceras de fragmentos**

(30) Prioridad:

15.06.2012 US 201261659986 P
15.04.2013 US 201313863072

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
25.03.2020

(73) Titular/es:

GOOGLE TECHNOLOGY HOLDINGS LLC (100.0%)
1600 Amphitheatre Parkway
Mountain View, CA 94043, US

(72) Inventor/es:

YU, YUE;
LOU, JIAN y
WANG, LIMIN

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Método y aparato para el tratamiento eficaz de cabeceras de fragmentos

ANTECEDENTES DE LA INVENCIÓN

1. Campo de la Invención

La presente invención se refiere a sistemas y métodos para codificar datos, y en particular, a un sistema y método para generar y tratar o procesar o cabeceras de fragmentos con datos codificados de video de alta eficiencia.

2. Descripción de la técnica relacionada

Se está produciendo un rápido crecimiento en las tecnologías asociadas con la generación, transmisión y reproducción de programas de medios. Estas tecnologías incluyen esquemas de codificación que permiten codificar versiones digitales de los programas de medios para comprimirlos a un tamaño mucho más pequeño y facilitar su transmisión, almacenamiento, recepción y reproducción. Estas tecnologías tienen aplicación en grabadores de video personales (PVR – Personal video Recorders, en inglés), video bajo demanda (VOD – Video On Demand, en inglés), ofertas de programas de medios de múltiples canales, interactividad, telefonía móvil y transmisión de programas de medios.

Sin compresión, los programas de medios digitales suelen ser demasiado grandes para ser transmitidos y/o almacenados con un coste comercialmente aceptable. No obstante, la compresión de tales programas ha hecho que la transmisión y el almacenamiento de tales programas de medios digitales no solo sea comercialmente factible, sino común.

Inicialmente, la transmisión de programas de medios incluía imágenes de baja a media resolución transmitidas a través de medios de transmisión de alto ancho de banda, tales como televisión por cable y satélite. No obstante, dicha transmisión ha evolucionado para incluir medios de transmisión de menor ancho de banda, tales como la transmisión por Internet a dispositivos fijos y móviles a través de redes informáticas, WiFi, TV móvil y redes de tercera y cuarta generación (3G y 4G). Además, tales transmisiones también han evolucionado para incluir programas de medios de alta definición, tales como la televisión de alta definición (HDTV – High Definition TV), que tienen un ancho de banda de transmisión y necesidades de almacenamiento importantes.

El estándar de codificación de codificación de video de alta eficiencia (HEVC – High Efficiency video Coding, en inglés) (o H.265) es el estándar de codificación más reciente promulgado por las organizaciones de estandarización MPEG de ISO/IEC. El estándar de codificación que precede al HEVC incluía el H.262/MPEG-2, y el estándar posterior H.264/MPEG-4 de codificación de video avanzada (AVC - Advanced Video Coding, en inglés). El estándar H.264/MPEG-4 ha reemplazado sustancialmente al H.262/MPEG-2 en muchas aplicaciones, incluida la televisión de alta definición (HD – High Definition, en inglés). HEVC admite resoluciones superiores a HD, incluso en realizaciones estéreo o de múltiples vistas, y es más adecuado para dispositivos móviles tales como ordenadores personales de tableta. Se puede encontrar más información relativa a HEVC en la publicación “Overview of the High Efficiency Video Coding (HEVC) Standard, por Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han y Thomas Weigand, IEEE Transaction on Circuits and Systems for Video Technology, diciembre de 2012.

Como en otros estándares de codificación, la estructura del flujo de bits y la sintaxis de los datos que cumplen con HEVC están estandarizados, de tal manera que cada descodificador que cumpla con el estándar producirá la misma salida cuando se le proporcione la misma entrada. Las características incorporadas en el estándar HEVC incluyen la definición y el procesamiento de un fragmento, uno o varios de los cuales, juntos, pueden comprender una de las imágenes en una secuencia de video. Una secuencia de video comprende una pluralidad de imágenes, y cada imagen puede comprender uno o varios más fragmentos. Los fragmentos incluyen fragmentos no dependientes y fragmentos dependientes. Un fragmento no dependiente (denominado simplemente, a continuación, en el presente documento, fragmento) es una estructura de datos que puede ser descodificada de manera independiente de otros fragmentos de la misma imagen en términos de codificación por entropía, predicción de señal y construcción de señal residual. Esta estructura de datos permite la resincronización de eventos en caso de pérdida de datos. Un “fragmento dependiente” es una estructura que permite que la información sobre el fragmento (tal como la relacionada con los recuadros dentro del fragmento o las entradas de frente de onda) sea llevada a la capa de red, haciendo que los datos estén disponibles para que un sistema procese más rápidamente fragmentos fragmentados. Los fragmentos dependientes son útiles, principalmente, para la codificación de bajo retardo.

La codificación y descodificación de fragmento s se realiza de acuerdo con la información incluida en la cabecera de un fragmento. La cabecera del fragmento incluye la sintaxis y la lógica para leer los indicadores y los datos que se utilizan para descodificar el fragmento. Puesto que cualquier secuencia de video determinada incluye, en general, miles de imágenes, y cada imagen puede contener uno o varios fragmentos, la sintaxis y la lógica utilizada en la cabecera pueden tener un impacto importante en la carga de procesamiento realizada para codificar y descodificar la secuencia de video. Por consiguiente, existe la necesidad de una sintaxis y una lógica de cabecera de fragmento que minimice el procesamiento necesario para descodificar y utilizar la cabecera del fragmento. La presente invención satisface esta necesidad.

Benjamin Bross; Woo-Jin Han; Jens-Rainer Ohm; Gary J Sullivan; Thomas Weigand: “High efficiency video coding (HEVC) text specification draft 7”, 12 de junio de 2012 (2012-06-12), páginas 1 a 258, Recuperado de Internet: URL:http://phenix.it-sudparis.eu/ict/doc end user/documents/9 Geneva/wg11/JCTVC-I1003-v5.zip [recuperado el 2013-07-03] describe una especificación de codificación de video que incluye una cabecera de fragmento.

Compendio de la Invención

Para abordar las necesidades descritas anteriormente, este documento describe un método que se puede utilizar en un sistema de procesamiento para descodificar una secuencia que comprende una pluralidad de imágenes, pudiendo estar dividida cada una de la pluralidad de imágenes en uno o varios fragmentos, siendo procesados cada uno de los uno o varios fragmentos, al menos en parte, de acuerdo con la cabecera del fragmento.

De acuerdo con un primer aspecto, se da a conocer un método para descodificar una secuencia que comprende una pluralidad de imágenes, tal como se describe en la reivindicación 1.

De acuerdo con un segundo aspecto, se da a conocer un método para descodificar una secuencia que comprende una pluralidad de imágenes, tal como se describe en la reivindicación 4.

De acuerdo con un tercer aspecto, se da a conocer un dispositivo de procesamiento para descodificar una secuencia que comprende una pluralidad de imágenes, tal como se describe en la reivindicación 7.

Las características, funciones y ventajas que se han explicado se pueden conseguir de manera independiente en diversas realizaciones de la presente invención, o pueden ser combinadas en otras realizaciones adicionales, detalles adicionales de las cuales se pueden ver con referencia a la siguiente descripción y dibujos.

Breve descripción de los dibujos

Con referencia, a continuación, a los dibujos en los que los números de referencia representan las partes correspondientes en toda su extensión:

La figura 1 es un diagrama que representa una realización a modo de ejemplo de un sistema de codificación - descodificación de video que puede ser utilizado para la transmisión y/o almacenamiento y la recuperación de información de audio y/o video;
la figura 2A es un diagrama de una realización de un sistema de códec en el que la información AV codificada es transmitida a otra ubicación y recibida en la misma;
la figura 2B es un diagrama que representa una realización a modo de ejemplo del sistema de códec en el que la información codificada es almacenada y, a continuación, recuperada, para su presentación, denominado a continuación, en el presente documento, sistema de almacenamiento de códec;
la figura 3 es un diagrama de bloques que ilustra una realización del codificador fuente;
la figura 4 es un diagrama que representa una imagen de la información AV, tal como una de las imágenes en la secuencia de imágenes;
la figura 5 es un diagrama que muestra una partición a modo de ejemplo de un bloque de árbol de codificación en unidades de codificación;
la figura 6 es un diagrama que ilustra una representación de un árbol cuaternario representativo y parámetros de datos para la partición de bloques del árbol de código que se muestra en la figura 5;
la figura 7 es un diagrama que ilustra la partición de una unidad de codificación en una o varias unidades de predicción;
la figura 8 es un diagrama que muestra una unidad de codificación dividida en cuatro unidades de predicción y un conjunto asociado de unidades de transformación;
la figura 9 es un diagrama que muestra el árbol de códigos RQT para las unidades de transformación asociadas con la unidad de codificación en el ejemplo de la figura 8;
la figura 10 es un diagrama que ilustra la predicción espacial de unidades de predicción;
la figura 11 es un diagrama que ilustra la predicción temporal;
la figura 12 es un diagrama que ilustra la utilización de predictores de vectores de movimiento (MVP – Motion Vector Predictors, en inglés);
la figura 13 ilustra un ejemplo de la utilización de las listas de imágenes de referencia;
la figura 14 es un diagrama que ilustra procesos realizados por el codificador de acuerdo con el estándar mencionado anteriormente;
la figura 15 representa la utilización de un indicador collocated_from_10_flag por el descodificador en la descodificación de acuerdo con el estándar HEVC emergente;
las figuras 16A a 16C son diagramas que presentan una sintaxis y lógica de cabecera de fragmento de referencia;
las figuras 17A y 17B presentan diagramas de sintaxis que ilustran una realización de una sintaxis de cabecera de fragmento mejorada;
las figuras 18A y 18B son diagramas que ilustran operaciones a modo de ejemplo que pueden ser llevadas a cabo de acuerdo con la cabecera del fragmento mostrada en las figuras 17A y 17B;

las figuras 19A y 19B presentan diagramas de sintaxis que ilustran otra realización de una sintaxis de cabecera de fragmento mejorada;
las figuras 20A y 20B presentan diagramas de sintaxis que ilustran otra realización de una sintaxis de cabecera de fragmento mejorada; y
la figura 21 ilustra un sistema de procesamiento a modo de ejemplo que podría ser utilizado para implementar las realizaciones de la invención.

DESCRIPCIÓN DETALLADA DE REALIZACIONES PREFERIDAS

En la siguiente descripción, se hace referencia a los dibujos adjuntos que forman parte de la misma, y que muestran, a modo de ilustración, varias realizaciones de la presente invención. Se comprende que se pueden utilizar otras realizaciones y se pueden realizar cambios estructurales sin apartarse del alcance de la presente invención.

Transmisión-recepción y almacenamiento de información audiovisual

La figura 1 es un diagrama que representa una realización a modo de ejemplo de un sistema 100 de codificación-descodificación (códec) de video que puede ser utilizado para la transmisión y/o almacenamiento y la recuperación de información de audio y/o video. El sistema de códec 100 comprende un sistema de codificación 104, que acepta información audiovisual (AV) 102 y procesa la información AV 102 para generar información AV codificada 106 (comprimida), y un sistema de descodificación 112, que procesa la información AV codificada 106 para producir la información AV recuperada 114. Puesto que los procesos de codificación y descodificación no son sin pérdidas, la información AV recuperada 114 no es idéntica a la información AV 102 inicial, sino que, con una selección sensata de los procesos y parámetros de codificación, las diferencias entre la información AV recuperada 114 y la información AV 102 no procesada son aceptables para la percepción humana.

La información AV codificada 106 es transmitida o almacena y recuperada, habitualmente, antes de la descodificación y presentación, tal como se realiza mediante el sistema de transmisión-recepción (transmisión y recepción) o de almacenamiento / recuperación 108. Las pérdidas de transmisión-recepción pueden ser importantes, pero las pérdidas de almacenamiento / recuperación son, habitualmente, mínimas o inexistentes, por lo tanto, la información AV transmitida-recibida 110 proporcionada al sistema de descodificación 112 es, habitualmente, igual o sustancialmente igual que la información AV codificada 106.

La figura 2A es un diagrama de una realización de un sistema de códec 200A en el que la información AV codificada 106 es transmitida a otra ubicación y recibida en la misma. Un segmento de transmisión 230 convierte una información AV 102 de entrada en una señal apropiada para la transmisión, y transmite la señal convertida en el canal de transmisión 212 al segmento de recepción 232. El segmento de recepción 232 recibe la señal transmitida y convierte la señal recibida en la información AV recuperada 114 para su presentación. Tal como se describió anteriormente, debido a las pérdidas y errores de codificación y transmisión, la información AV recuperada 114 puede ser de menor calidad que la información AV 102 que fue proporcionada al segmento de transmisión 230. No obstante, se pueden incluir sistemas de corrección de errores para reducir o eliminar tales errores. Por ejemplo, la información AV codificada 106 puede ser codificada con corrección de errores hacia delante (FEC – Forward Error Correction, en inglés) mediante la adición de información redundante, y dicha información redundante puede ser utilizada para identificar y eliminar errores en el segmento de recepción 230.

El segmento de transmisión 102 comprende uno o varios codificadores fuente 202 para codificar múltiples fuentes de información AV 102. El codificador fuente 202 codifica la información AV 102 principalmente para propósitos de compresión, para producir la información AV codificada 106, y puede incluir, por ejemplo, un procesador e instrucciones de almacenamiento de memoria relacionadas que implementan un códec tal como MPEG-1, MPEG-2, MPEG-4 AVC/H.264, HEVC o códec similar, tal como se describe más adelante con más detalle.

El sistema de códec 200A puede incluir, asimismo, elementos opcionales, indicados por las líneas discontinuas en la figura 2A. Estos elementos opcionales incluyen un codificador multiplex de video 204, un controlador de codificación 208 y un descodificador multiplexación de video 218. El codificador multiplex de video 204 opcional multiplexa la información AV codificada 106 de una pluralidad asociada de codificadores fuente 202 de acuerdo con uno o varios parámetros proporcionados por el controlador de codificación 208 opcional. Dicha multiplexación se realiza, habitualmente, en el dominio del tiempo, y está basada en paquetes de datos.

En una realización, el codificador multiplex de video 204 comprende un multiplexador estadístico, que combina la información AV codificada 106 de una pluralidad de codificadores fuente 202 para minimizar el ancho de banda necesario para la transmisión. Esto es posible, puesto que la velocidad de bits instantánea de la información AV codificada 106 de cada codificador fuente 202 puede variar mucho con el tiempo según el contenido de la información AV 102. Por ejemplo, escenas que tienen una gran cantidad de detalles y movimiento (por ejemplo, eventos deportivos) se codifican, habitualmente, en velocidades de bits más altas que las escenas con poco movimiento o detalles (por ejemplo, diálogo vertical). Puesto que cada codificador fuente 202 puede producir información con una velocidad de bits instantánea alta, mientras que otro codificador fuente 202 produce información con una velocidad de bits instantánea baja, y puesto que el controlador de codificación 208 puede ordenar a los codificadores fuente 202 que codifiquen la información AV 106 de acuerdo con ciertos parámetros de rendimiento que afectan a la velocidad de bits instantánea, las señales de cada uno de los codificadores fuente 106 (cada una

con una velocidad de bits instantánea variable temporalmente) pueden ser combinadas de manera óptima para minimizar la velocidad de bits instantánea de la secuencia multiplexada 205.

Tal como se describió anteriormente, el codificador fuente 202 y el codificador multiplex de video 204 pueden ser controlados, opcionalmente, por un controlador de codificación 208 para minimizar la velocidad de bits instantánea de la señal de video combinada. En una realización, esto se consigue utilizando información de la memoria intermedia del transmisor 206 que almacena temporalmente la señal de video codificada y puede indicar el llenado de la memoria intermedia del transmisor 206. Esto permite que la codificación realizada en el codificador 202 de fuente o en el codificador multiplex de video 204 sea una función del almacenamiento restante en la memoria intermedia del transmisor 206.

El segmento de transmisión 230 puede comprender, asimismo, un codificador de transmisión 210, que codifica adicionalmente la señal de video para su transmisión al fragmento de recepción 232. La codificación de transmisión puede incluir, por ejemplo, la codificación FEC mencionada anteriormente y/o la codificación en un esquema de multiplexación para el medio de transmisión elegido. Por ejemplo, si la transmisión es por satélite o transmisores terrestres, el codificador de transmisión 114 puede codificar la señal en una constelación de señales antes de la transmisión mediante modulación de amplitud en cuadratura (QAM – Quadrature Amplitude Modulation, en inglés) o una técnica de modulación similar. Asimismo, si la señal de video codificada debe ser transmitida en tiempo real a través de un dispositivo de protocolo de Internet y de la Internet, la transmisión codifica la señal de acuerdo con el protocolo apropiado. Además, si la señal codificada va a ser transmitida a mediante telefonía móvil, se utiliza el protocolo de codificación apropiado, tal como se describe más adelante con más detalle.

El segmento de recepción 232 comprende un descodificador de transmisión 214 para recibir la señal que fue codificada por el codificador de transmisión 210 utilizando un esquema de descodificación complementario al esquema de codificación utilizado en el codificador de transmisión 214. La señal recibida descodificada puede ser almacenada temporalmente mediante la memoria intermedia del receptor 216, y si la señal recibida comprende múltiples señales de video, la señal recibida es descodificada por el descodificador multiplex de video 218 para extraer la señal de video de interés de las señales de video multiplexadas por el codificador multiplex de video 204. Finalmente, la señal de video de interés es descodificada por el descodificador fuente 220 utilizando un esquema de descodificación o códec complementario al códec utilizado por el codificador fuente 202 para codificar la información AV 102.

En una realización, los datos transmitidos comprenden un flujo de video en paquetes transmitido desde un servidor (que representa el segmento de transmisión 230) a un cliente (que representa el segmento de recepción 232). En este caso, el codificador de transmisión 210 puede empaquetar los datos e incrustar unidades de capa abstracta de red (NAL – Network Abstract Layer, en inglés) en paquetes de red. Las unidades de NAL definen un contenedor de datos que tiene elementos principales y codificados, y puede corresponder a un videograma o a otro fragmento de datos de video.

Los datos comprimidos a transmitir pueden ser empaquetados y transmitidos a través del canal de transmisión 212, que puede incluir una red de área amplia (WAN – Wide Area Network, en inglés) o una red de área local (LAN – Local Area Network, en inglés). Dicha red puede comprender, por ejemplo, una red inalámbrica tal como WiFi, una red Ethernet, una red de Internet o una red mixta compuesta por varias redes diferentes. Dicha comunicación puede verse afectada a través de un protocolo de comunicación, por ejemplo, el protocolo de transporte en tiempo real (RTP – Real-time Transport Protocol, en inglés), el protocolo de datagramas de usuario (UDP – User Datagram Protocol, en inglés) o cualquier otro tipo de protocolo de comunicación. Se pueden utilizar diferentes métodos de empaquetamiento para cada unidad de capa abstracta de red (NAL) del flujo de bits. En un caso, el tamaño de una unidad de NAL es menor que el tamaño de la máxima unidad de transporte (MTU – Maximum Transport Unit, en inglés) correspondiente al tamaño de paquete más grande que puede ser transmitido a través de la red sin ser fragmentada. En este caso, la unidad de NAL está integrada en un único paquete de red. En otro caso, se incluyen múltiples unidades de NAL completas en un solo paquete de red. En un tercer caso, una unidad de NAL puede ser demasiado grande para ser transmitida en un solo paquete de red y, por lo tanto, se divide en varias unidades de NAL fragmentadas con cada unidad de NAL fragmentada que se transmite en un paquete de red individual. Las unidades de NAL fragmentadas son enviadas, habitualmente, de manera consecutiva para fines de descodificación.

El segmento de recepción 232 recibe los datos empaquetados y reconstituye las unidades de NAL del paquete de red. Para unidades de NAL fragmentadas, el cliente concatena los datos de las unidades de NAL fragmentadas para reconstruir la unidad de NAL original. El cliente 232 descodifica el flujo de datos recibido y reconstruido y reproduce las imágenes de video en un dispositivo de visualización y los datos de audio mediante un altavoz.

La figura 2B es un diagrama que representa una realización a modo de ejemplo del sistema de códec en el que la información codificada es almacenada y, a continuación, recuperada, para su presentación, denominado a continuación en el presente documento, sistema de almacenamiento de códec 200B. Esta realización puede ser utilizada, por ejemplo, para almacenar localmente información en un grabador de video digital (DVR – Digital Video Recorder, en inglés), una unidad rápida, disco duro o dispositivo similar. En esta realización, la información AV 102 está codificada en origen por el codificador de origen 202, opcionalmente almacenado en la memoria intermedia de

almacenamiento 234 antes del almacenamiento en un dispositivo de almacenamiento 236. El dispositivo de almacenamiento 236 puede almacenar la señal de video temporalmente o durante un período de tiempo extendido, y puede comprender un disco duro, unidad de memoria rápida, RAM o ROM. La información AV almacenada es recuperada, y, opcionalmente, almacenada, en la memoria temporal de recuperación 238 y descodificada por el descodificador fuente 220.

La figura 2C es otro diagrama que representa un sistema de distribución de contenido a modo de ejemplo 200C, que comprende un sistema de codificación o codificador 202 y un sistema de descodificación o descodificador 220 que pueden ser utilizados para transmitir y recibir datos de HEVC. En algunas realizaciones, el sistema de codificación 202 puede comprender una interfaz de entrada 256, un controlador 241, un contador 242 una memoria de fotogramas 243, una unidad de codificación 244, una memoria intermedia del transmisor 267 y una interfaz de salida 257. El sistema de descodificación 220 puede comprender una memoria intermedia del receptor 259, una unidad de descodificación 260, una memoria de fotogramas 261 y un controlador 267. El sistema de codificación 202 y el sistema de descodificación 220 pueden estar acoplados entre sí a través de una ruta de transmisión que puede transportar un flujo de bits comprimido. El controlador 241 del sistema de codificación 202 puede controlar la cantidad de datos a transmitir en función de la capacidad de la memoria intermedia del transmisor 267 o la memoria intermedia del receptor 259, y puede incluir otros parámetros tales como la cantidad de datos por unidad de tiempo. El controlador 241 puede controlar la unidad de codificación 244 para evitar el fallo de una operación de descodificación de señal recibida del sistema de descodificación 220. El controlador 241 puede ser un procesador o incluir, a modo de ejemplo no limitativo, un microordenador que tiene un procesador, una memoria de acceso aleatorio y una memoria de solo lectura.

Las imágenes fuente 246 suministradas desde, a modo de ejemplo no limitativo, un proveedor de contenido, pueden incluir una secuencia de fotogramas de video que incluye imágenes fuente en una secuencia de video. Las imágenes fuente 246 pueden ser descomprimidas o comprimidas. Si las imágenes fuente 246 no están comprimidas, el sistema de codificación 202 puede tener una función de codificación. Si las imágenes fuente 246 están comprimidas, el sistema de codificación 202 puede tener una función de transcodificación. Las unidades de codificación pueden ser derivadas de las imágenes fuente utilizando el controlador 241. La memoria de fotogramas 243 puede tener una primera área, que puede ser utilizada para almacenar los fotogramas entrantes de las imágenes fuente 246, y una segunda área, que puede ser utilizada para leer los fotogramas y emitirlos a la unidad de codificación 244. El controlador 241 puede emitir una señal de control de cambio de área 249 a la memoria de fotogramas 243. La señal de control de cambio de área 249 puede indicar si se utilizará la primera área o la segunda área.

El controlador 241 puede emitir una señal de control de codificación 250 a la unidad de codificación 244. La señal de control de codificación 250 puede hacer que la unidad de codificación 202 inicie una operación de codificación, tal como preparar las unidades de codificación basadas en una imagen fuente. En respuesta a la señal de control de codificación 250 del controlador 241, la unidad de codificación 244 puede comenzar a leer las unidades de codificación preparadas en un proceso de codificación de alta eficiencia, tal como un proceso de codificación de predicción o un proceso de codificación de transformación que procesa las unidades de codificación preparadas generando datos de compresión de video basados en las imágenes fuente asociadas con las unidades de codificación.

La unidad de codificación 244 puede empaquetar los datos de compresión de video generados en un flujo elemental empaquetado (PES – Packetized Elementary Stream, en inglés) que incluye paquetes de video. La unidad de codificación 244 puede asignar (map, en inglés) los paquetes de video en una señal de video codificada 248 utilizando información de control y una marca de tiempo de presentación (PTS – Program Time Stamp, en inglés) y la señal de video codificada 248 puede ser transmitida a la memoria intermedia del transmisor 267.

La señal de video codificada 248, que incluye los datos de compresión de video generados, se puede almacenar en la memoria intermedia del transmisor 267. El contador de cantidad de información 242 se puede incrementar para indicar la cantidad total de datos en la memoria intermedia del transmisor 267. A medida que los datos son recuperados y eliminados de la memoria intermedia, el contador 242 puede ser decrementado para reflejar la cantidad de datos en la memoria intermedia del transmisor 267. La señal de información del área ocupada 253 puede ser transmitida al contador 242 para indicar si se han añadido o eliminado datos de la unidad de codificación 244 a la memoria intermedia del transmisor 267 para que el contador 242 pueda ser incrementado o decrementado. El controlador 241 puede controlar la producción de paquetes de video producidos por la unidad de codificación 244 en base a la información del área ocupada 253 que puede ser comunicada para anticipar, evitar, prevenir y/o detectar que ocurra un desbordamiento o subflujo en la memoria intermedia del transmisor 267.

El contador de cantidad de información 242 puede ser reiniciado en respuesta a una señal 254 preestablecida generada y emitida por el controlador 241. Después de que el contador de cantidad de información 242 ha sido reiniciado, puede contar los datos emitidos por la unidad de codificación 244 y obtener la cantidad de datos de compresión de video y/o de paquetes de video que han sido generados. El contador de cantidad de información 242 puede suministrar al controlador 241 una señal de cantidad de información 255 representativa de la cantidad de

información obtenida. El controlador 241 puede controlar la unidad de codificación 244 para que no haya desbordamiento en la memoria intermedia del transmisor 267.

En algunas realizaciones, el sistema de descodificación 220 puede comprender una interfaz de entrada 266, una memoria intermedia del receptor 259, un controlador 267, una memoria de fotogramas 261, una unidad de descodificación 260 y una interfaz de salida 267. La memoria intermedia del receptor 259 del sistema de descodificación 220 puede almacenar temporalmente el flujo de bits comprimido, incluidos los datos de compresión de video recibidos y los paquetes de video basados en las imágenes fuente de las imágenes fuente 246. El sistema de descodificación 220 puede leer la información de control y la información de la marca de tiempo de presentación asociada con los paquetes video en los datos recibidos y emitir una señal de número de fotogramas 263 que puede ser aplicada al controlador 220. El controlador 267 puede supervisar el número contado de fotogramas en un intervalo predeterminado. A modo de ejemplo no limitativo, el controlador 267 puede supervisar el número de fotogramas contados cada vez que la unidad de descodificación 260 completa una operación de descodificación.

En algunas realizaciones, cuando la señal de número de fotogramas 263 indica que la memoria intermedia del receptor 259 está a una capacidad predeterminada, el controlador 267 puede emitir una señal de inicio de descodificación 264 a la unidad de descodificación 260. Cuando la señal del número de fotogramas 263 indica que la memoria intermedia del receptor 259 está a menos de una capacidad predeterminada, el nivel de control 267 puede esperar a que ocurra una situación en la que el número de fotogramas contados sea igual a una cantidad predeterminada. El controlador 267 puede emitir la señal de inicio de descodificación 263 cuando se produce la situación. A modo de ejemplo no limitativo, el controlador 267 puede emitir la señal de inicio de descodificación 264 cuando la señal de número de fotogramas 263 indica que la memoria intermedia del receptor 259 está a la capacidad predeterminada. Los paquetes de video codificados y los datos de compresión de video pueden ser descodificados en un orden monótono (es decir, aumentando o disminuyendo) en función de las marcas de tiempo de presentación asociadas con los paquetes de video codificados.

En respuesta a la señal de inicio de descodificación 264, la unidad de descodificación 260 puede descodificar datos que equivalen a una imagen asociada con un fotograma, y datos de video comprimidos asociados con la imagen asociada con paquetes de video desde la memoria intermedia del receptor 259. La unidad de descodificación 260 puede escribir una señal de video descodificada 269 en la memoria de fotogramas 261. La memoria de fotogramas 261 puede tener una primera área en la que se escribe la señal de video descodificada, y una segunda área utilizada para leer imágenes descodificadas 262 en la interfaz de salida 267.

En diversas realizaciones, el sistema de codificación 202 puede ser incorporado o asociado de otra manera con un transcodificador o un aparato de codificación en una cabecera, y el sistema de descodificación 220 puede ser incorporado o asociado de otra manera con un dispositivo de más abajo, tal como un dispositivo móvil, un descodificador o un transcodificador.

Codificación / descodificación de fuente

Tal como se describió anteriormente, los codificadores 202 emplean algoritmos de compresión para generar flujos de bits y/o archivos de menor tamaño que las secuencias de video originales en la información AV 102. Dicha compresión se hace posible reduciendo las redundancias espaciales y temporales en las secuencias originales.

Los codificadores 202 de la técnica anterior incluyen aquellos que cumplen con el estándar de compresión de video H.264/MPEG-4 AVC (“Advanced Video Coding”) desarrollado entre el “Video Coding Expert Group” (VCEG) de la ITU y el “Moving Picture Expert Group” (MPEG) de la ISO, en particular en la forma de la publicación “Advanced Video Coding for Generic Audiovisual Services” (marzo de 2005), que se incorpora por referencia en el presente documento.

HEVC “High Efficiency Video Coding (a veces conocido como H.265) se espera que reemplace al AVC H.264/MPEG-4. HEVC presenta nuevas herramientas y entidades de codificación que son generalizaciones de las entidades de codificación definidas en H.264/AVC, tal como se describe más adelante con más detalle. CS39543 / CS39549 / CS39892.

La figura 3 es un diagrama de bloques que ilustra una forma de realización del codificador fuente 202. El codificador fuente 202 acepta información AV 102 y utiliza el muestreo mediante el muestreador 302 de la información AV 102 para producir una secuencia 303 de sucesivas imágenes digitales o imágenes, cada una de las cuales tiene una pluralidad de píxeles. Una imagen puede comprender un fotograma o un campo, en el que un fotograma es una imagen completa capturada durante un intervalo de tiempo conocido, y un campo es el conjunto de líneas de exploración de número pare o impar que componen una imagen parcial.

El muestreador 302 produce una secuencia de imagen sin comprimir 303. Cada imagen digital puede estar representada por una o varias matrices que tienen una pluralidad de coeficientes que representan información sobre los píxeles, que, juntos, comprenden la imagen. El valor de un píxel puede corresponder a la luminancia o a otra información. En el caso de que varios componentes estén asociados con cada píxel (por ejemplo, componentes rojo

– verde - azul o componentes de luminancia - crominancia), cada uno de estos componentes puede ser procesado por separado.

Las imágenes pueden estar segmentadas en “fragmentos”, que pueden comprender una parte de la imagen o pueden comprender la imagen completa. En el estándar H.264, estos fragmentos están divididos en entidades de codificación denominadas macrobloques (en general, bloques del tamaño de 16 píxeles x 16 píxeles) y cada macrobloque, a su vez, puede estar dividido en diferentes tamaños de bloques de datos 102, por ejemplo, 4x4, 4x8, 8x4, 8x8, 8x16, 16x8. HEVC expande y generaliza la noción de la entidad de codificación más allá de la del macrobloque.

Entidades de codificación de HEVC: CTU. CU. PU y TU

Como otros estándares de codificación de video, HEVC es un esquema de codificación predictivo espacial y temporal híbrido basado en bloques. No obstante, HEVC presenta nuevas funciones de codificación que no están incluidas con el estándar H.264/AVC. Estas entidades de codificación incluyen (1) Bloque de árbol de codificación (CTU – Coding Tree Units, en inglés), unidades de codificación (CU – Coding Units, en inglés), las unidades predictivas (PU – Predictive Units, en inglés) y las unidades de transformación (TU – Transform Units, en inglés), y se describen adicionalmente a continuación.

La figura 4 es un diagrama que representa una imagen 400 de información AV 102, como una de las imágenes en la secuencia de imágenes 303. La imagen 400 está dividida espacialmente en bloques cuadrados no superpuestos conocidos como unidades de árbol de codificación o CTU 402. A diferencia de H.264 y los estándares de codificación de video anteriores, en los que la unidad de codificación básica es un macrobloque de 16x16 píxeles, la CTU 402 es la unidad de codificación básica de HEVC, y puede ser tan grande como 128x128 píxeles. Tal como se muestra en la figura 4, las CTU 402 están referenciadas habitualmente dentro de la imagen 400 en un orden análogo a una exploración progresiva.

Cada CTU 402 puede ser dividida, a su vez, iterativamente, en unidades de codificación de tamaño variable más pequeñas descritas por una descomposición de “árbol cuaternario” que se describe más adelante. Las unidades de codificación son regiones formadas en la imagen a las que se aplican y transmiten parámetros de codificación similares en el flujo de bits 314.

La figura 5 es un diagrama que muestra una partición a modo de ejemplo de una CTU 402 en unidades de codificación (CU) tales como la unidad de codificación 502A y 502B (denominadas alternativamente a continuación, en el presente documento, unidad o unidades de codificación 502). Una única CTU 402 se puede dividir en cuatro CU 502, tales como la CU 502A, teniendo cada una un cuarto del tamaño de la CTU 402. Cada una de estas CU 502A se puede dividir en cuatro CU 502B más pequeñas de un cuarto del tamaño de la CU 502A inicial.

La división de las CTU 402 en CU 502A y en CU 502B más pequeñas se describe mediante parámetros de datos “árbol cuaternario” (por ejemplo, indicadores o bits) que están codificados en el flujo de bits de salida 314 junto con los datos codificados como sobrecoste conocidos como sintaxis.

La figura 6 es un diagrama que ilustra una representación de un árbol cuaternario 600 representativo y parámetros de datos para la partición CTU 402 mostrada en la figura 5. El árbol cuaternario 600 comprende una pluralidad de nodos que incluyen el primer nodo 602A en un nivel jerárquico y el segundo nodo 602B en un nivel jerárquico inferior (en adelante, los nodos de árbol cuaternario se pueden denominar alternativamente “nodos” 602). A cada nodo 602 de un árbol cuaternario, se le asigna un “indicador de división” o bit “1” si el nodo 602 está dividido, además, en subnodos; en caso contrario, se le asigna un bit “0”.

Por ejemplo, la partición CTU 402 ilustrada en la figura 5 se puede representar mediante el árbol cuaternario 600 presentado previamente en la figura 6, que incluye un indicador de división “1” asociado con el nodo 602A al nivel de la CU 502 superior (lo que indica que hay 4 nodos adicionales en un nivel jerárquico inferior). El árbol cuaternario 600 ilustrado incluye, asimismo, un indicador de división “1” “asociado con el nodo 602B al nivel de la CU 502 intermedia, para indicar que esta CU también está dividida en otras cuatro CU 502 al nivel de la siguiente (inferior) de CU. El codificador fuente 202 puede restringir los tamaños mínimos y máximos de la CU 502, cambiando de este modo la profundidad máxima posible de la división de la CU 502.

El codificador 202 genera información AV codificada 106 en forma de un flujo de bits 314 que incluye una primera porción que tiene datos codificados para las CU 502 y una segunda porción que incluye un sobrecoste conocido como elementos de sintaxis. Los datos codificados incluyen datos correspondientes a las CU 502 codificadas (es decir, los residuos codificados junto con sus vectores de movimiento asociados, predictores o residuos relacionados tal como se describe más adelante). La segunda porción incluye elementos de sintaxis que pueden representar parámetros de codificación que no corresponden directamente a los datos codificados de los bloques. Por ejemplo, los elementos de impuestos sincronizados pueden comprender una dirección e identificación de la CU 502 en la imagen, un parámetro de cuantificación, una indicación del modo de codificación Inter / Intra elegido, el árbol cuaternario 600 u otra información.

Las CU 502 corresponden a elementos de codificación elementales e incluyen dos subunidades relacionadas: las unidades de predicción (PU) y las unidades de transformación (TU), las cuales tienen un tamaño máximo igual a la CU 502 correspondiente.

La figura 7 es un diagrama que ilustra la partición de una CU 502 en una o varias PU 702. Una PU 702 corresponde a una CU 502 dividida y se utiliza para predecir valores de píxeles para los tipos de intra-imagen o inter-imágenes. Las PU 702 son una extensión de la partición de H.264/AVC para la estimación de movimiento, y se definen para cada CU 502 que no está subdividida en otras CU ("indicador de división" = 0). En cada hoja 604 del árbol cuaternario 600, una CU 502 final (nivel inferior) de 2Nx2N puede poseer uno de los cuatro patrones posibles de PU: 2Nx2N (702A), 2NxN (702B), Nx2N (702C) y NxN (702D)), tal como se muestra en la figura 7.

Una CU 502 puede tener un código predictivo espacial o temporal. Si una CU 502 está codificada en modo "intra", cada PU 702 de la CU 502 puede tener su propia dirección de predicción espacial e información de imagen tal como se describe más adelante. Además, en el modo "intra", la PU 702 de la CU 502 puede depender de otra CU 502, porque puede utilizar una cercana en el espacio, que está en otra CU. Si una CU 502 está codificada en modo "inter", cada PU 702 de la CU 502 puede tener sus propios vectores de movimiento y las imágenes de referencia asociadas tal como se describe más adelante.

La figura 8 es un diagrama que muestra una CU 502 dividida en cuatro PU 702 y un conjunto asociado de unidades de transformación (TU) 802. Las TU 802 se utilizan para representar las unidades elementales que son transformadas espacialmente mediante una DCT (Transformada discreta del coseno – Discrete Cosine Transform, en inglés). El tamaño y la ubicación de cada bloque de transformación TU 802 dentro de una CU 502 se describe mediante un árbol cuaternario "residual" (RQT – Residual QuadTree, en inglés) que se ilustra más adelante.

La figura 9 es un diagrama que muestra el RQT 900 para la TU 802 para la CU 502 en el ejemplo de la figura 8. Se debe observar que el "1" en el primer nodo 902A del RQT 900 indica que hay cuatro ramas y, que el "1" en el segundo nodo 902B en el nivel jerárquico inferior contiguo indica que el nodo indicado tiene, además, cuatro ramas. Los datos que describen el RQT 900 también son codificados y transmitidos como un sobrecoste en el flujo de bits 314.

Los parámetros de codificación de una secuencia de video pueden ser almacenados en unidades de NAL dedicadas llamadas conjuntos de parámetros. Se pueden emplear dos tipos de conjuntos de parámetros de unidades de NAL. El primer tipo de conjunto de parámetros se conoce como conjunto de parámetros de secuencia (SPS – Sequence Parameter Set, en inglés) y comprende una unidad de NAL que incluye parámetros que no cambian durante toda la secuencia de video. Habitualmente, un SPS maneja el perfil de codificación, el tamaño de los fotogramas de video y otros parámetros. El segundo tipo de conjunto de parámetros se conoce como conjunto de parámetros de imagen (PPS – Picture Parameter Set, en inglés), y codifica diferentes valores que pueden cambiar de una imagen a otra.

Predicción espacial y temporal

Una de las técnicas utilizadas para comprimir un flujo de bits 314 es prescindir del almacenamiento de los valores de los píxeles y, en su lugar, predecir los valores de los píxeles utilizando un proceso que puede ser repetido en el descodificador 220 y almacenar o transmitir la diferencia entre los valores de los píxeles predichos y los valores de los píxeles reales (conocido como el residuo). Mientras el descodificador 220 pueda calcular los mismos valores de píxel predichos a partir de la información proporcionada, los valores de imagen reales se pueden recuperar sumando los residuos a los valores predichos. La misma técnica se puede utilizar, asimismo, para comprimir otros datos.

Con referencia de nuevo a la figura 3, cada PU 702 de la CU 502 que se procesa es proporcionada a un módulo predictor 307. El módulo predictor 307 predice los valores de las PU 702 en base a la información en las PU 702 cercanas en el mismo fotograma (predicción intra-fotogramas, lo que se realiza mediante el predictor espacial 324), y la información de las PU 702 en fotogramas temporalmente próximos (predicción entre fotogramas, que se realiza mediante el predictor temporal 330). No obstante, la predicción temporal no siempre se basa en una PU situada en el mismo sitio, ya que las PU situadas en el mismo sitio están definidas para esetar situadas en un fotograma de referencia / no referencia que tenga las mismas coordenadas x e y que la PU 702 actual. Estas técnicas aprovechan las dependencias espaciales y temporales entre las PU 702.

Por lo tanto, las unidades codificadas se pueden clasificar para incluir dos tipos: (1) unidades no predichas temporalmente y (2) unidades predichas temporalmente. Las unidades predichas no temporalmente se predicen utilizando el fotograma actual, incluidas las PU 702 contiguas o cercanas intra-fotograma (por ejemplo, la predicción intra-fotograma), y son generadas por el predictor espacial 324. Las unidades predichas temporalmente son predichas a partir de una imagen temporal (por ejemplo, los fotogramas P) o predichas a partir de al menos dos imágenes de referencia temporalmente delante y/o detrás (es decir, los fotogramas B).

Predicción espacial

La figura 10 es un diagrama que ilustra la predicción espacial de las PU 702. Una imagen puede comprender una PU 702 y otras PU 1 a 4 espacialmente próximas, incluida la PU 702N próxima. El predictor espacial 324 predice el

bloque actual (por ejemplo, el bloque C de la figura 10) por medio de una predicción “intra-fotograma” que utiliza la PU 702 de otros bloques de píxeles ya codificados de la imagen actual.

El predictor 324 localiza una PU próxima (por ejemplo, la PU 1, 2, 3 o 4 de la figura 10) que es apropiada para la codificación espacial, y determina una dirección de predicción angular a esa PU próxima. En HEVC, se pueden considerar 35 direcciones, por lo que cada PU puede tener una de las 35 direcciones asociadas, incluidas horizontal, vertical, diagonal a 45 grados, diagonal a 135 grados, DC, etc. La dirección de predicción espacial de la PU se indica en la sintaxis.

Con referencia de nuevo al predictor espacial 324 de la figura 3, esta PU situada próxima se utiliza para calcular una PU 704 residual (e) como la diferencia entre los píxeles de la PU 702N próxima y la PU 702 actual, utilizando el elemento 305. El resultado es un elemento 1006 de PU intra-predicha que comprende una dirección de predicción 1002 y la PU residual intra-predicha 1004. La dirección de predicción 1002 puede ser codificada deduciendo la dirección de las PU espacialmente próximas y las dependencias espaciales de la imagen, permitiendo que la velocidad de codificación del modo de dirección de intra-predicción se reduzca.

Predicción temporal

La figura 11 es un diagrama que ilustra la predicción temporal. La predicción temporal considera la información de imágenes o fotogramas temporalmente contiguas, tal como la imagen anterior, imagen i-1.

En general, la predicción temporal incluye la predicción única (tipo P), que predice la PU 702 refiriéndose a un área de referencia desde una sola imagen de referencia, y la predicción múltiple (tipo B), que predice la PU refiriéndose a dos áreas de referencia desde una o dos imágenes de referencia. Las imágenes de referencia son imágenes en la secuencia de video que ya han sido codificadas y luego reconstruidas (mediante descodificación).

El predictor temporal 330 identifica, en una o varias de estas áreas de referencia (una para el tipo P o varias para el tipo B), áreas de píxeles en un fotograma temporal cercano para que puedan ser utilizadas como predictores de esta PU 702 actual. En el caso de que se utilicen predictores de varias áreas (tipo B), pueden ser fusionados para generar una sola predicción.

El área de referencia 1102 se identifica en el fotograma de referencia mediante un vector de movimiento (MV – Motion Vector, en inglés) 1104 que define el desplazamiento entre la PU 702 actual en el fotograma actual (imagen i) y el área de referencia 1102 (refIdx) en el fotograma de referencia (imagen i-1). Una PU en una imagen B puede tener hasta dos MV. Tanto la información de MV como la de refIdx están incluidas en la sintaxis del flujo de bits de HEVC.

Con referencia de nuevo a la figura 3, una diferencia entre los valores de píxel entre el área de referencia 1102 y la PU 702 actual se puede calcular mediante el elemento 305 seleccionado por el conmutador 306. Esta diferencia se denomina el residuo de la PU 1106 inter-predicha. Al final del proceso de predicción temporal o inter-fotogramas, la PU 1006 actual está compuesta por un vector de movimiento MV 1104 y un residuo 1106.

No obstante, tal como se describió anteriormente, una técnica para comprimir datos es generar valores predichos para los datos utilizando medios repetibles por el descodificador 220, calculando la diferencia entre los valores predichos y reales de los datos (el residuo) y transmitiendo el residuo para su descodificación. Mientras el descodificador 220 pueda reproducir los valores predichos, los valores residuales se pueden utilizar para determinar los valores reales.

Esta técnica se puede aplicar a los MV 1104 utilizados en la predicción temporal generando una predicción del MV 1104, calculando una diferencia entre el MV 1104 real y el MV 1104 predicho (un residuo) y transmitiendo el residuo de MV en el flujo de bits 314. Siempre que el descodificador 220 pueda reproducir el MV 1104 predicho, el MV 1104 real se puede calcular a partir del residuo. HEVC calcula un MV predicho para cada PU 702 utilizando la correlación espacial del movimiento entre las PU 702 próximas.

La figura 12 es un diagrama que ilustra la utilización de predictores del vector de movimiento (MVP) en HEVC. Los predictores del vector de movimiento V_1, V_2 y V_3 se toman de los MV 1104 de una pluralidad de bloques 1, 2 y 3 situados cerca o contiguos al bloque a codificar (C). Puesto que estos vectores se refieren a vectores de movimiento de bloques espacialmente próximos dentro del mismo fotograma temporal y se pueden utilizar para predecir el vector de movimiento del bloque a codificar, estos vectores se conocen como predictores del movimiento espacial.

La figura 12 ilustra, asimismo, el predictor del vector de movimiento temporal V_T, que es el vector de movimiento del bloque C’ situado en el mismo sitio en una imagen previamente descodificada (en orden de descodificación) de la secuencia (por ejemplo, el bloque de imagen i-1 situado en la misma posición espacial que el bloque que se está codificando (bloque C de la imagen i).

Los componentes de los predictores del vector de movimiento espacial V_1, V_2 y V_3 y el predictor del vector de movimiento temporal V_T se pueden utilizar para generar un predictor de vector de movimiento medio V_M. En HEVC,

se pueden tomar los tres predictores del vector de movimiento espacial tal como se muestra en la figura 12, es decir, desde el bloque situado a la izquierda del bloque a codificar (V_1), desde el bloque situado encima (V_3) y desde uno de los bloques situados en las esquinas respectivas del bloque a codificar (V_2), de acuerdo con una regla predeterminada de disponibilidad. Esta técnica de selección de predictores de MV se conoce como predicción avanzada de vectores de movimiento (AMVP – Advanced Motion Vector Prediction, en inglés).

Por lo tanto, se obtienen a pluralidad (habitualmente cinco) de predictores de MV (MVP) candidatos que tienen predictores espaciales (por ejemplo, V_1, V_2 y V_3) y predictor o predictores temporales V_T. Para reducir el sobrecoste de señalizar el predictor del vector de movimiento en el flujo de bits, el conjunto de predictores del vector de movimiento se puede reducir eliminando datos para vectores de movimiento duplicados (por ejemplo, MV que tienen el mismo valor que otros MV pueden ser eliminados de los candidatos).

El codificador 202 puede seleccionar un “mejor” predictor del vector de movimiento entre los candidatos, y calcular un residuo del predictor del vector de movimiento como la diferencia entre el predictor del vector de movimiento seleccionado y el vector de movimiento real, y transmitir el residuo del predictor del vector de movimiento en el flujo de bits 314. Para llevar a cabo esta operación, el descodificador 220 debe almacenar el vector de movimiento real para su utilización posterior (aunque no se transmite en el flujo de bits 314). Los bits o indicadores de señalización se incluyen en el flujo de bits 314 para especificar qué residuo de MV se calculó a partir del predictor del vector de movimiento normalizado, y, son utilizados, a continuación, por el descodificador para recuperar el vector de movimiento. Estos bits o indicadores se describen adicionalmente a continuación.

Con referencia de nuevo a la figura 3, los residuos intra-predichos 1004 y los residuos inter-predichos 1106 obtenidos del proceso de predicción espacial (intra) o temporal (inter) son transformados a continuación mediante el módulo de transformación 308 en las unidades de transformación (TU) 802 descritas anteriormente. Una TU 802 puede ser dividida, adicionalmente, en TU más pequeñas utilizando la descomposición de RQT descrita anteriormente con respecto a la figura 9. En HEVC, en general, se utilizan 2 o 3 niveles de descomposición, y los tamaños de transformación autorizados son de 32x32,16x16, 8x8 y 4x4. Tal como se describió anteriormente, la transformación se deriva de acuerdo con una transformada discreta del coseno (DCT) o una transformada discreta del seno (DST – Discrete Sine Transform, en inglés).

A continuación, los coeficientes residuales transformados son cuantificados por el cuantificador 310. La cuantificación realiza una función muy importante en la compresión de datos. En HEVC, la cuantificación convierte los coeficientes de transformación de alta precisión en un número finito de valores posibles. Aunque la cuantificación permite una gran cantidad de compresión, la cuantificación es una operación con pérdidas, y la pérdida por cuantificación no se puede recuperar.

Los coeficientes del residuo transformado cuantificado son codificados, a continuación, por medio de un codificador por entropía 312 y, a continuación, son insertados en el flujo de bits 310 comprimido como parte de los datos útiles que codifican las imágenes de la información AV. Los elementos de sintaxis de codificación también pueden ser codificados utilizando dependencias espaciales entre elementos de sintaxis, para aumentar la eficiencia de la codificación. HEVC ofrece codificación aritmética binaria adaptativa al contexto (CABAC – Context-Adaptive Binary Arithmetic Coding, en inglés). Asimismo, se pueden utilizar otras formas de codificación, por entropía o aritmética.

Con el fin de calcular los predictores utilizados anteriormente, el codificador 202 descodifica las PU 702 ya codificadas utilizando el bucle 315 de “descodificación”, que incluye los elementos 316, 318, 320, 322, 328. Este bucle 315 de descodificación reconstruye las PU y las imágenes a partir de los residuos transformados cuantificados.

Los coeficientes E de los residuos transformados cuantificados son proporcionados al descuantificador 316, que aplica la operación inversa a la del cuantificador 310 para producir coeficientes transformados descuantificados de la PU del residuo (*E*’) 708. Los datos descuantificados 708 son proporcionados, a continuación, a un transformador inverso 318, que aplica la inversa de la transformación aplicada por el módulo de transformación 308 para generar coeficientes residuales reconstruidos de la PU (*e*’) 710.

Los coeficientes reconstruidos de la PU del residuo 710 son añadidos, a continuación, a los coeficientes de la PU predicha (x’) 702’ correspondiente seleccionados de la PU intra-predicha 1004 y la PU inter-predicha 1106 por el selector 306. Por ejemplo, si el residuo reconstruido proviene del proceso de codificación “intra” del predictor espacial 324, el predictor “intra” (x’) es añadido a este residuo para recuperar una PU reconstruida (x’’) 712 correspondiente a la PU 702 original modificada por las pérdidas resultantes de una transformación, por ejemplo, en este caso, de las operaciones de cuantificación. Si el residuo 710 proviene de un proceso de codificación “inter” del predictor temporal 330, las áreas señaladas por los vectores de movimiento actuales (estas áreas pertenecen a las imágenes de referencia almacenadas en la memoria intermedia de referencia 328 referidas por los índices de imágenes actuales) son fusionadas y, a continuación, añadidas a este residuo descodificado. De esta manera, la PU 702 original es modificada por las pérdidas resultantes de las operaciones de cuantificación.

En la medida en que el codificador 202 utiliza técnicas de predicción de vectores de movimiento análogas a las técnicas de predicción de imagen descritas anteriormente, el vector de movimiento puede ser almacenado utilizando

la memoria intermedia de vectores de movimiento 329 para su utilización en subtramas posteriores en el tiempo. Tal como se describe más adelante con más detalle, un indicador puede ser configurado y transferido en la sintaxis para indicar que el vector de movimiento para el fotograma actualmente descodificado debe ser utilizado, al menos para el fotograma codificado posteriormente, en lugar de reemplazar el contenido de la memoria intermedia 329 de MV con el MV para el fotograma actual.

Un filtro de bucle 322 es aplicado a la señal reconstruida (x'') 712 para reducir los efectos creados por la cuantificación intensa de los residuos obtenidos y para mejorar la calidad de la señal. El filtro de bucle 322 puede comprender, por ejemplo, un filtro de desbloqueo, para suavizar los bordes entre las PU para atenuar visualmente las altas frecuencias creadas por el proceso de codificación, y un filtro lineal, que se aplica después de que se hayan descodificado todas las PU para una imagen, con el fin de minimizar la suma de la diferencia cuadrada (SSD – Sum of the Square Difference, en inglés) con la imagen original. El proceso de filtrado lineal se realiza fotograma a fotograma, y utiliza varios píxeles alrededor del píxel a filtrar, y utiliza, asimismo, dependencias espaciales entre píxeles del fotograma. Los coeficientes del filtro lineal pueden ser codificados y transmitidos en una cabecera del flujo de bits, habitualmente una cabecera de imagen o fragmento.

Las imágenes filtradas, también conocidas como imágenes reconstruidas, son almacenadas como imágenes de referencia de la memoria intermedia de imágenes de referencia 328 para permitir que tengan lugar predicciones "Inter" posteriores durante la compresión de las imágenes posteriores de la secuencia de video actual.

Sintaxis de imagen de referencia

Tal como se describió anteriormente, para reducir errores y mejorar la compresión, HEVC permite la utilización de varias imágenes de referencia para la estimación y la compensación del movimiento de la imagen actual. Dada una PU 702 actual en una imagen actual, la PU 1102 situada en el mismo sitio para un fragmento particular reside en una imagen de referencia / no referencia próxima asociada. Por ejemplo, en la figura 12, la PU 1102 situada en el mismo sitio para la PU 702 actual en la imagen (i) reside en la imagen de referencia próxima asociada (i-1). Los mejores predictores "inter" o temporales de la PU 702 actual son seleccionados en algunas de las múltiples imágenes de referencia / no referencia, que pueden estar basadas en imágenes temporalmente anteriores a la imagen actual o posteriores a la misma en orden de visualización (predicción hacia atrás y hacia delante, respectivamente).

Para HEVC, el índice de imágenes de referencia se define mediante listas de imágenes de referencia que se describen en la sintaxis de fragmento. La predicción hacia delante está definida por list_015 (RefPicList0), y la predicción hacia atrás está definida por list_1 (RefPicList1), y tanto la lista 0 como la lista 1 pueden contener múltiples imágenes de referencia antes o después de la imagen actual en el orden de visualización.

La figura 13 ilustra un ejemplo de la utilización de las listas de imágenes de referencia. Considérense las imágenes 0, 2, 4, 5, 6, 8 y 10 que se muestran en la figura 13, en la que los números de cada imagen denotan el orden de visualización y la imagen actual es la imagen 5. En este caso, las imágenes de referencia de list_0 con índices de imagen de referencia ascendentes y que comienzan en con un índice igual a cero son 4, 2, 0, 6, 8 y 10, y las imágenes de referencia de list_1 con índices de imagen de referencia ascendentes y que comienzan con un índice igual a cero son 6, 8, 10, 4, 2 y 0. Un fragmento en el que la predicción de compensación de movimiento está restringida a la predicción de list_0 se denomina predictivo, o fragmento P. Las imágenes situadas en el mismo sitio se indican mediante la utilización del índice collocated_ref_idx en la HEVC. Un fragmento para el que la predicción de compensación de movimiento incluye más de una imagen de referencia es un fragmento de doble predicción o fragmento B. Para los fragmentos B, la predicción de compensación de movimiento puede incluir imágenes de referencia de la predicción de list_1, así como de list_0.

Por lo tanto, una PU 1102 situada en el mismo sitio está dispuesta en una imagen de referencia especificada en list_0 o en list_1. Se utiliza un indicador (collocated_from_10_flag) para especificar si la partición situada en el mismo sitio debe ser derivada de list_0 o list_1 para un tipo de fragmento en particular. Cada una de las imágenes de referencia está asociada, asimismo, con un vector de movimiento.

El almacenamiento y la recuperación de imágenes de referencia y vectores de movimiento relacionados para el estándar de HEVC emergente se expresan en el párrafo 8.4.1.2.9 de Benjamin Bross, Woo-Jin Han, Jens-Rainer Ohm, Gary J. Sullivan, Thomas Wiegand, "WD4: Working Draft 4 of High-Efficiency Video Coding", Joint Collaborative Team on video Coding (JCT-VC) de ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, JCTVC-F803_d5, 6ª Reunión: Turín, IT, 14-22 de julio de 2011 (incorporada en el presente documento por referencia).

De acuerdo con el estándar, si el slice_type es igual a B y el collocated_from_10_flag es 0, la variable collocated_ref_idx especifica la imagen de referencia como la imagen que contiene la partición situada en el mismo sitio según lo especificado por RefPicList1. En caso contrario (slice_type es igual a B y collocated_from_10_flag es igual a 1 o slice_type es igual a P), la variable collocated_ref_idx especifica la imagen de referencia como la imagen que contiene la partición situada en el mismo sitio según lo especificado por RefPicList0.

La figura 14 es un diagrama que ilustra los procesos realizados por el codificador 202 de acuerdo con el estándar mencionado anteriormente. El bloque 1402 determina si la imagen actual es una imagen de referencia para otra imagen. En caso contrario, no es necesario almacenar la imagen de referencia o la información del vector de movimiento. Si la imagen actual es una imagen de referencia para otra imagen, el bloque 1504 determina si la "otra" imagen es una imagen de tipo P o de tipo B. Si la imagen es una imagen de tipo P, el procesamiento pasa a los bloques 1410, que configuran colloc_from_10_flag a uno y almacenan la imagen de referencia y el vector de movimiento en list0. Si la "otra imagen" es una imagen de tipo B, el bloque 1406 dirige el procesamiento a los bloques 1408 y 1410 si la imagen de referencia deseada va a ser almacenada en list_0, y a los bloques 1412 y 1414 si la imagen de referencia y el vector de movimiento deseados van a ser almacenados en list_1. Esta decisión se puede basar en si es deseable seleccionar imágenes de referencia de una imagen temporal anterior o posterior. La selección de las múltiples imágenes de referencia posibles se determina de acuerdo con el índice collocated_ref_idx.

La figura 15 representa la utilización de un collocated_from_10_flag por el descodificador 220 en la descodificación de acuerdo con el estándar HEVC anterior. El bloque 1502 determina si el tipo de fragmento actual que se está calculando es un tipo intra o I. Dichos fragmentos no utilizan fragmentos temporalmente cercanos en el proceso de codificación / descodificación y, por lo tanto, no hay necesidad de encontrar una imagen de referencia temporalmente cercana. Si el tipo de fragmento no es de tipo I, el bloque 1504 determina si el fragmento es un fragmento B. Si el fragmento no es de tipo B, es un fragmento de tipo P, y la imagen de referencia que contiene la partición situada en el mismo sitio se encuentra en list_0, de acuerdo con el valor de collocated_ref_idx. Si el fragmento es de tipo B, collocated_from_10_flag determina si la imagen de referencia se encuentra en list_0 o en list_1. Como indica el índice, la imagen situada en el mismo sitio se define como la imagen de referencia que tiene el collocated_ref_idx indicado en list_ 0 o en la list_1, según el tipo de fragmento (tipo B o tipo P) y el valor de collocated_from_10_flag. En una realización de HEVC, la primera imagen de referencia (la imagen de referencia que tiene el índice [0] tal como se muestra en la figura 13 es seleccionada como la imagen situada en el mismo sitio).

Sintaxis de cabecera de fragmento de referencia

Las figuras 16A a 16C son diagramas que presentan una lógica y una sintaxis de la cabecera de fragmento de referencia. En los diagramas, la sangría del texto indica la estructura lógica de la sintaxis, en la que el delímetro "| |" representa un "O" lógico, "&&" representa un "Y" lógico "!" representa un inverso lógico o complemento. Además, si una instrucción de condición lógica (por ejemplo, "si" la instrucción es verdadera, las operaciones con sangría de la instrucción "si" lógica (y encerradas entre llaves "{}") son ejecutadas; en caso contrario, el procesamiento continúa a la siguiente instrucción lógica.

Volviendo primero a la figura 16A, la sintaxis del procesamiento del fragmento difiere dependiendo de si el fragmento es el primero de una pluralidad de fragmentos en una imagen, o si no es el primer fragmento en la imagen. En consecuencia, la cabecera del fragmento comprende un primer fragmento en el indicador de imagen (first_slice_in_pic_flag) que se lee. Esto se ilustra en la sintaxis 1602.

Tal como se describió anteriormente, el estándar HEVC incluye una pluralidad de tipos de unidades de NAL que incluyen un conjunto de parámetros de video (VPS), un conjunto de parámetros de secuencia (SPS), que presenta parámetros para una secuencia de imágenes y un conjunto de parámetros de imagen (PPS), que describe parámetros para una imagen en particular. Asimismo, se lee un identificador del conjunto de parámetros de imagen (pic_parameter_set_id). Si el fragmento no es el primer fragmento en la imagen, se lee la dirección del fragmento. Esto se ilustra en la sintaxis 1606.

Tal como se describió anteriormente, los fragmentos pueden incluir fragmentos no dependientes o fragmentos dependientes, y la sintaxis de la cabecera del fragmento permite la deshabilitación o la habilitación de la utilización de fragmentos dependientes juntos. La siguiente lógica utiliza un indicador de lectura previa (dependent_slice_enabled_flag), que indica que los fragmentos dependientes están habilitados, y first_slice_in_pic_flag, para determinar si se debe leer el dependente_slice_flag. Se debe observar que, si el fragmento es el primer fragmento en la imagen, el indicador del fragmento dependiente para este fragmento no se lee, ya que el fragmento no puede ser un fragmento dependiente en tales circunstancias. Si el fragmento no es un fragmento dependiente, la lógica que sigue lee el tipo de fragmento y otros parámetros que se utilizan en el procesamiento posterior para todos los tipos de fragmento (I, P y B). El procesamiento adicional que se muestra en la sintaxis 1612 también se lleva a cabo.

Pasando a continuación a la figura 16B, la sintaxis 1614 lee una variable aps_id si el adaptive_loop_filter_enabled_flag leído en el conjunto de parámetros de secuencia (SPS) está habilitado.

La sintaxis 1615 incluye una instrucción condicional que prueba si los datos de slice_type leídos anteriormente en la cabecera del fragmento indican si el tipo de fragmento es o P o B. Si el tipo de fragmento no es ni P ni B, el procesamiento es encaminado para determinar si leer la lista de imágenes de referencia (ref_pic_list_modification) se lee o no, tal como se explica a continuación con referencia a la sintaxis 1619. Si el tipo de fragmento es un P o B, la lógica utiliza un sps_temporal_mvp_enable_flag que se leyó como parte de la sintaxis de la cabecera de SPS para determinar si el fragmento puede ser descodificado utilizando un predictor de vector de movimiento temporal. Si se configura el indicador, lo que indica que el predictor del vector de movimiento temporal puede estar habilitado, se lee

un indicador que describe si el predictor del vector de movimiento temporal está permitido para la imagen que contiene el fragmento (pic_temporal_mvp_enable_flag).

Independientemente del estado del sps_temporal_mvp_enable_flag, se lee otro indicador (num_ref_idx_active_override_flag) tal como se muestra en la sintaxis 1617. Este indicador indica si un parámetro (num_ref_inx_10_active_minus1) que describe el índice máximo de la lista de imágenes de referencia máximas para list_0 (tipo P) u otro parámetro (num_ref_idx_11_active_minus1) que describe el índice máximo de la lista de imágenes de referencia para list_1 (tipo B) están presentes en la cabecera del fragmento. Si el (num_ref_idx_active_override_flag) resulta ser positivo, se lee el parámetro num_ref_inx_10_active_minus1, y si el fragmento es un fragmento de tipo B, también se lee el parámetro num_ref_inx_11_active_minus1, tal como se muestra en la sintaxis 1618

HEVC permite que la referencia de la imagen de referencia sea modificada en el proceso de codificación. Sin tener en cuenta el tipo de fragmento (ya que las operaciones que siguen no están dentro del condicional si, la sintaxis 1615 prueba si el fragmento es un tipo P o un tipo B), se lee un indicador previamente leído (lists_modification_present_flag), en una realización, del PPS. Si el lists_modification_present_flag resulta ser un 1 lógico, se ejecuta la sintaxis de modificaciones de la lista de referencia (ref_pic_list_modification ()).

Esta información es utilizada por la sintaxis del ref_pic_list_modification para leer, en base al tipo de fragmento, un indicador (reference_pic_list_modification_flag_10) que identifica si el fragmento fue codificado de acuerdo con una lista de imágenes de referencia implícita (si el indicador es un cero lógico o no existe) o si la lista de imágenes de referencia para la lista de imágenes de referencia asociada con el fragmento debe ser definida de manera explícita (si el indicador es un 1 lógico), en cuyo caso se leen las entradas de la lista para la lista de imágenes de referencia. Tal como se describe más adelante con más detalle, la sintaxis de la referencia ref_pic_list_modification () incluye instrucciones condicionales lógicas basadas en el tipo de fragmento, que son simplificadas en las soluciones descritas a continuación.

A continuación, la lógica de la cabecera del fragmento nuevamente determina si el fragmento considerado es un fragmento de tipo B y, si es así, lee un mvd_11_zero_flag. El mvd_11_zero_flag no es aplicable a fragmentos de tipo P e indica si la estructura de la sintaxis de codificación de diferencia de vector de movimiento utilizada con fragmentos de tipo B se analiza o no. Esto se muestra en la sintaxis 1620.

Tal como se describió anteriormente con referencia al codificador 312 ilustrado en la figura 3, HEVC implementa la codificación adaptativa al contexto, como la codificación aritmética binaria adaptativa al contexto, o CABAC. CABAC es una forma de codificación por entropía que codifica símbolos binarios utilizando modelos de probabilidad. Un símbolo de valor no binario (tal como un coeficiente de la unidad de transformación o un vector de movimiento) es binarizado, o convertido en un código binario antes de la codificación aritmética. Las etapas se repiten para cada bit (o “bin”) del símbolo binarizado.

Un modelo de contexto es un modelo de probabilidad para uno o varios contenedores del símbolo binarizado. Este modelo puede ser elegido entre una pluralidad de modelos disponibles dependiendo de las estadísticas de los símbolos de datos codificados recientemente. El modelo de contexto almacena la probabilidad de que cada bin sea “1” o “0”. A continuación, un codificador aritmético codifica cada bin de acuerdo con el modelo de probabilidad seleccionado.

Una variable de contexto es una variable especificada para el proceso de descodificación aritmética binaria adaptativa de un contenedor mediante una ecuación que contiene contenedores recientemente descodificados. Un cabac_init_flag especifica el método para determinar la tabla de inicialización utilizada en el proceso de inicialización para las variables de contexto. El valor de cabac_init_flag es 0 a 1, inclusivo. Cuando cabac_init_flag no está presente, se infiere que es 0.

Volviendo a la figura 16B, sin tener en cuenta el resultado de la lógica que determina si el fragmento fue un fragmento inter-predicho (tipo P o B) tal como se ha descrito en la sintaxis 1615 anterior, la lógica de la cabecera del fragmento verifica si el tipo de fragmento no es un tipo I y un indicador de señalización (cabac_init_present_flag) que indica si un indicador de inicialización de contexto variable (cabac_init_flag) está presente en la cabecera del fragmento y debe ser leído. Si el tipo de fragmento no es de tipo I y el indicador de señalización indica que el indicador de inicialización de la variable de contexto está presente en la cabecera del fragmento, se lee el indicador de inicialización de la variable de contexto. El indicador de variable de inicialización de la variable de contexto especifica el método para determinar la tabla de inicialización utilizada en el proceso de inicialización de la variable de contexto. Esto se muestra en la sintaxis 1622.

Sin tener en cuenta el resultado de la sintaxis 1615 que determina si el fragmento era un fragmento inter-predicho (tipo P o B), la lógica de la cabecera del fragmento lee una variable (slice_qp_delta) que describe el valor inicial para un parámetro de cuantificación que se utilizará en los bloques de codificación de datos. Este valor inicial se utiliza hasta que se modifica en la unidad de codificación. Esto se ilustra con la sintaxis 1624.

Tal como se describió anteriormente, el filtro de bucle 322 del codificador / descodificador puede comprender, por ejemplo, un filtro de desbloqueo, para suavizar los bordes entre las PU, para atenuar visualmente las altas frecuencias creadas por el proceso de codificación, y un filtro lineal, que se aplica después de que todas las PU para una imagen han sido descodificadas para minimizar la suma de la diferencia cuadrada (SSD) con la imagen original. El proceso de filtrado lineal se realiza fotograma a fotograma y utiliza varios píxeles alrededor del píxel para ser filtrado, y también utiliza dependencias espaciales entre los píxeles del fotograma. Los coeficientes de filtro lineal pueden ser codificados y transmitidos en una cabecera del flujo de bits, habitualmente una cabecera de imagen o fragmento.

Volviendo a la figura 16B, la lógica de cabecera de fragmento ejecuta la lógica de filtro de desbloqueo, tal como se ilustra con respecto a la sintaxis 1626. Esta etapa se lleva a cabo sin tener en cuenta el resultado de la lógica que determina si el fragmento era un fragmento inter-predicho (tipo P o B) tal como se describe en la sintaxis 1615 indicada anteriormente. En primer lugar, la lógica de la cabecera del fragmento determina si un control de filtro de desbloqueo está habilitado, verificando el estado de un indicador de control (deblocking_filter_control_present_flag) en el conjunto de parámetros de imagen (PPS). Si el indicador resulta ser verdadero, la lógica realiza una verificación para determinar si el filtro de desbloqueo se anula verificando otro indicador (deblocking_filter_override_enabled_flag) que indica que la cabecera del fragmento para las imágenes que hacen referencia al PPS tiene un deblocking_filter_override_flag. Si este filtro está habilitado, se lee un indicador (deblocking_filter_override_flag) que indica que se debe anular el filtro de desbloqueo. La lógica determina, a continuación, si el filtro de anulación de desbloqueo está configurado, y si es así, lee un indicador de nivel de cabecera de fragmento (slice_header_disable_deblocking_filter_flag) (eso indica si el filtro de desbloqueo debe ser deshabilitado. Si el slice_header_disable_deblocking_filter_flag no está configurado, la lógica de la cabecera del fragmento lee los datos de beta_offset_div2 y tc_offset_div2, que especifican los desplazamientos por defecto de los parámetros de desbloqueo.

A continuación, la lógica de la cabecera del fragmento realiza operaciones relacionadas con la determinación de la ubicación de la imagen situada en el mismo sitio utilizada para el predictor del vector de movimiento temporal. La cabecera de fragmento primero comprueba si el predictor del vector de movimiento temporal está habilitado en un nivel de fragmento / imagen comprobando pic_temporal_mvp_enable_flag, tal como se muestra en la sintaxis 1628. Si el indicador no está configurado, el procesamiento se dirige a la predicción ponderada que se analiza más adelante con más detalle. Si se configura el indicador, la lógica de la cabecera del fragmento determina si el tipo de fragmento es B, tal como se muestra en la sintaxis 1630. Si el tipo de fragmento es B, la lógica de la cabecera del fragmento lee collocated_from_10_flag, tal como se muestra en la sintaxis 1632. A continuación, la lógica determina si el tipo de fragmento no es de tipo I y (1) la combinación lógica de collocated_from_10_flag y num_ref_idx_10_active_minus1 es mayor que cero o (2) la combinación lógica del inverso de collocated_from_10_flag y num_ref_idx_active_minus1 es mayor que cero). Si cualquiera de estas posibilidades resulta ser un 1 lógico (o Verdadero), se lee el índice de referencia situado en el mismo sitio (collocated_ref_idx), tal como se muestra en la sintaxis 1634.

HEVC y los estándares de codificación previos permitieron una operación de escalado y desplazamiento que se aplica a las señales de predicción de una manera conocida como predicción ponderada. Mientras que H.264/MPEG-4 AVC admitía la predicción ponderada tanto temporalmente implícita como explícita, en HEVC, solo se aplica la predicción ponderada explícita, escalando y desplazando la predicción con valores enviados de manera explícita por el codificador. La profundidad de bits de la predicción se configura a la profundidad de bits original de las muestras de referencia. En el caso de predicción única, el valor de predicción interpolado (y posiblemente ponderado) es redondeado, desplazado a la derecha y recortado para tener la profundidad de bits original. En el caso de la predicción doble, los valores de predicción interpolados (y posiblemente ponderados) de dos PB son añadidos en primer lugar y, a continuación, redondeados, desplazados a la derecha y recortados.

En los estándares de codificación anteriores, se requieren hasta tres etapas de operaciones de redondeo para obtener cada muestra de predicción (para muestras situadas en posiciones de un cuarto de muestra). De este modo, si se utiliza doble predicción, el número total de operaciones de redondeo es entonces siete, en el peor de los casos. En HEVC, como máximo se necesitan dos operaciones de redondeo para obtener cada muestra situada en las posiciones de un cuarto de muestra; de este modo, cinco operaciones de redondeo son suficientes, en el peor de los casos, cuando se utiliza la doble predicción. Además, en el caso más común, en el que la profundidad de bits B es de 8 bits, el número total de operaciones de redondeo, en el peor de los casos, se reduce aún más a tres. Debido al menor número de operaciones de redondeo, el error de redondeo acumulado disminuye y se permite una mayor flexibilidad con respecto a la manera de realizar las operaciones necesarias en el descodificador.

Volviendo a la figura 16B, la lógica de la cabecera del fragmento utiliza el tipo de fragmento y los indicadores que indican si se debe habilitar la predicción ponderada o la doble predicción. El weighted_pred_flag está configurado en el 0 lógico para indicar que la predicción ponderada no se aplica a los fragmentos P, y está configurado en el 1 lógico para indicar que la predicción ponderada se aplica a los fragmentos P. El weighted_bipred_flag está configurado en el 0 lógico para especificar que la predicción ponderada predeterminada se aplica a los fragmentos B y, configurado en el 1 lógico, especifica que la predicción ponderada se aplica a los fragmentos B. Por lo tanto, la lógica de la cabecera del fragmento incluye lógica para leer la tabla de predicción de ponderación

((pred_weight_table ()) si el weighted_pred_flag está configurado en el 1 lógico y el tipo de fragmento es P, o si el weighted_bipred_flag está configurado en el 1 lógico y el tipo de fragmento es B, tal como se muestra en la sintaxis 1636.

Se puede especificar un número máximo de candidatos de predicción de vectores de movimiento que son admitidos en el fragmento. En la lógica de la cabecera del fragmento, esto se especifica como la diferencia entre el número “5” y el número máximo, y se conoce como el five_minus_max_num_merge_cand. En la siguiente lógica de la cabecera del fragmento, si el tipo de fragmento es un tipo P o un tipo B, se lee el five_minus_max_num_merge_cand, tal como se muestra en la sintaxis 1638. Puesto que el número máximo de candidatos es habitualmente cinco, el número leído es habitualmente cero.

El filtrado de bucle adaptativo a nivel de fragmento se ha incluido en algunas ediciones del estándar HEVC. En dichas ediciones, la lógica de la cabecera del fragmento detecta si el filtro de bucle adaptativo está habilitado comprobando el adaptive_loop_filter_enabled_flag. Si el adaptive_loop_filter_enabled_flag está configurado en el 1 lógico, se lee slice_adaptive_loop_filter_flag, lo que indica si el filtrado de bucle adaptativo se debe realizar a nivel de fragmento. Esto se muestra en la sintaxis 1640.

El slice_adaptive_loop_filter_flag se utiliza, por lo tanto, para determinar si los parámetros de filtrado de bucle adaptativo (alf_param ()) se leen cuando los coeficientes de filtrado de bucle adaptativo están en el fragmento (indicado por un alf_coef_in_slice_flag) o diferentes parámetros (alf_cu_control_param ()) cuando los coeficientes de filtrado adaptativo no están en el fragmento, tal como se muestra en la sintaxis 1640.

Finalmente, HEVC permite que las operaciones de filtrado en bucle se realicen a través de los límites izquierdo y superior del fragmento actual. Las ediciones anteriores de la cabecera de fragmento HEVC incluían un slice_loop_filter_across_slices_enabled_flag, que cuando está configurado en 1 especifica que estas operaciones de filtrado en bucle (incluido el filtro de desbloqueo y el filtro de desplazamiento adaptativo de muestra) se realizan a través de los límites izquierdo y superior del fragmento actual; de lo contrario, las operaciones en bucle en bucle no se aplican a través de los límites izquierdo y superior del fragmento actual. La lógica de sintaxis 1642 lee el slice_loop_filter_across_slices_enabled_flag si la función está habilitada en un nivel de secuencia (por ejemplo, el seq_loop_filter_across_slices_enabled_flag está configurado y, bien el slice _adaptive_loop_filter_enable_flag, o bien el slice_sample_adaptive_offset_flag está configurado, o bien el disable_deblocking_filter_flag no está configurado, tal como se muestra en la sintaxis 1642. La lógica de sintaxis de la cabecera del fragmento restante 1644 se refiere a la utilización de recuadros o extensiones de la cabecera del fragmento.

Sintaxis de cabecera de fragmento modificada

El diseño de la cabecera del fragmento de referencia incluye sintaxis y llamadas a funciones, incluso bajo las mismas condiciones lógicas (por ejemplo, bajo el mismo tipo de fragmento), desplegado en diferentes ubicaciones en la cabecera del fragmento. Si bien dicho diseño es necesario cuando la funcionalidad involucrada incluye dependencias o relaciones causales que se deben mantener, pueden resultar en una verificación innecesaria de la condición lógica. Esto es particularmente pernicioso cuando se utiliza en un nivel de cabecera de fragmento, ya que un programa de medios habitual comprende una gran cantidad de fragmentos y, por lo tanto, una gran cantidad de cabeceras de fragmento. A continuación, se presentan sintaxis de cabecera de fragmento modificadas que reducen o eliminan bits y pruebas lógicas en exceso cuando no son necesarios.

Primera solución: agregar parámetro de cuantificación de fragmento inicial

Valores y parámetros de desbloqueo después de las condiciones lógicas de tipo P y B agrupadas

Las figuras 17A y 17B presentan diagramas de sintaxis que ilustran una realización de una sintaxis de cabecera de fragmento mejorada. En esta realización, todas las llamadas a funciones correspondientes realizadas para los fragmentos de tipo P y B están agrupadas, y el slice_qp_delta y la correspondiente sintaxis de desbloqueo están dispuestas después de dichas llamadas a funciones.

De manera más específica, la sintaxis 1616 a 1618 de la cabecera del fragmento de referencia fue ejecutada si el tipo de fragmento se probó como tipo P o tipo B; en caso contrario, la lógica saltó a la sintaxis 1619 y las instrucciones que seguían. En la primera solución, se incorpora lógica adicional dentro de la instrucción condicional, y se simplifica para eliminar llamadas innecesarias a funciones. Específicamente, en la primera solución, la lógica del predictor del vector de movimiento temporal de la sintaxis 1616 y la lógica de anulación del índice de referencia de la sintaxis 1617 a 1618 permanecen sin cambios. No obstante, también se ejecuta otra lógica si la instrucción condicional de la sintaxis 1615 resulta ser verdadero. Esto incluye la sintaxis 1619 relacionada con el ref_pic_list_modification (), la sintaxis relacionada con el mvd_11_zero_flag 1620, la sintaxis relacionada con el indicador de inicialización CABAC 1622, la sintaxis 1628 a 1634 del predictor del vector de movimiento temporal para leer el índice de referencia situado en el mismo sitio, y la sintaxis 1636 de la tabla de predicción ponderada.

Además, las llamadas innecesarias a funciones se eliminan. Por ejemplo, en la estructura de la cabecera del fragmento de referencia, la sintaxis reference_pic_list_modification () incluye una prueba condicional para determinar si el fragmento es un fragmento P o B. Mover la sintaxis 1619 bajo la prueba condicional 1615 permite eliminar la

prueba en el reference_pic_list_modification (). Además, la sintaxis 1622 relacionada con CABAC de referencia de la cabecera del fragmento incluye una prueba condicional para determinar si el fragmento es un fragmento de tipo I. Esa prueba condicional se elimina disponiendo la sintaxis 1622 relacionada con CABAC dentro de la lógica de la sintaxis 1615. Además, la sintaxis 1638 para la lectura del parámetro five_minus_max_num_merge_cand ya no requiere su propia lógica anterior para garantizar que se lea solo si el tipo de fragmento es P o B (como fue el caso en la sintaxis de referencia mostrada en la figura 16B), puesto que esta operación leída se lleva a cabo a continuación bajo la condición lógica en la sintaxis 1615.

Las figuras 17A y 17B son diagramas de una sintaxis de cabecera de fragmento mejorada a modo de ejemplo. Las figuras 17A y 17B se explicarán con referencia a las figuras 18A y 18B, que presentan un diagrama de flujo de la lógica presentada en las figuras 17A y 17B.

Pasando, en primer lugar, a la figura 18A, se realiza una determinación en el bloque 1802 con respecto a si el fragmento es un fragmento inter-predicho o no. Esto se muestra, por ejemplo, mediante la sintaxis 1701, que prueba si el tipo de fragmento es tipo P o tipo B. En otra realización, esto puede ser implementado mediante la prueba lógica de si el tipo de fragmento no es de tipo I.

Si el fragmento no es un fragmento inter-predicho, la lógica pasa a la sintaxis 1716 de procesamiento de slice_qp_delta de la figura 17B. Si el fragmento es un fragmento inter-predicho (es decir, un fragmento de tipo P o un fragmento de tipo B, pero no un fragmento de tipo I), la lógica continúa al bloque 1804, que comprueba si el sps_temporal_mvp_enable_flag indica que la secuencia en la que la imagen está dispuesta puede ser codificada utilizando el predictor del vector de movimiento temporal y, si es así, se lee un pic_temporal_mvp_enable_flag en el bloque 1806. Esto se ilustra, asimismo, en la sintaxis 1702. El procesamiento pasa al bloque 1808, que lee el indicador de anulación de imagen de referencia (num_ref_idx_active_override_flag), ilustrado, asimismo, en la sintaxis 1703. El bloque 1810 prueba si la cabecera del fragmento incluye datos que describen el índice máximo de referencia y, si es así, se leen los datos que describen este índice máximo, tal como se muestra en el bloque 1812. Tal como se muestra más adelante en la sintaxis 1704, el parámetro num_ref_idx_10_active_minus1 se lee si el fragmento es un fragmento P, y ambos parámetros num_ref_idx_10_active_minus1 y num_ref_idx_11_active_minus1 se leen si el fragmento es un fragmento B.

El bloque 1814 determina si las imágenes de referencia están definidas de manera implícita. Si las imágenes de referencia no están definidas de manera implícita, sino que se obtienen de una lista, el procesamiento pasa al bloque 1816, en el que se leen los datos de modificación de la lista de imágenes de referencia, y el procesamiento pasa al bloque 1818. En caso contrario, el procesamiento pasa al bloque 1818 sin leer los datos de modificación de la lista de imágenes de referencia. Esto también se ilustra mediante la sintaxis 1705 mostrada en la figura 17A. Se debe observar que el procesamiento que se muestra en la sintaxis 1705 se puede hacer sin referencia (por ejemplo, utilizando una lógica condicional) del tipo de fragmento, puesto que esta operación es pertinente para todos los fragmentos de tipo P y de tipo B, y esta condición se determinó en la sintaxis 1701.

La información que indica una estructura de las diferencias del vector de movimiento para la lista L1 utilizada en la predicción de desplazamiento del movimiento se lee si el fragmento es un fragmento B. Esto se ilustra en los bloques 1818 y 1810 de la figura 18A y mediante la sintaxis 1706 de la figura 17A, que lee el mvd_11_zero_flag si el fragmento es un fragmento B.

La información asociada con la inicialización de las variables de contexto utilizadas en la codificación de entropía se lee, a continuación, de acuerdo con un indicador, tal como se muestra en el bloque 1822 de la figura 18A. La sintaxis ilustrativa para llevar a cabo esta operación se muestra en la figura 17A como sintaxis 1708. Se debe observar de nuevo que el procesamiento que se muestra en la sintaxis 1708 se puede llevar a cabo sin referencia (por ejemplo, utilizando una lógica condicional) al tipo de fragmento, puesto que esta operación es pertinente para todos los fragmentos de tipo P y los fragmentos de tipo B, y esta condición se determinó en la sintaxis 1702, y no es necesario volver a evaluarla.

Si el fragmento se codificó con al menos un predictor de vector de movimiento temporal, y el fragmento es de tipo B, se lee un indicador (collocated_from_10_flag) que controla la lectura de un índice (collocated_ref_idx) que indica qué imagen de referencia se utiliza como imagen situada en el mismo sitio en 10 u 11. Esto se ilustra en los bloques 1824 a 1826 de la figura 18A y la sintaxis 1710 ilustrada en la figura 17A.

La predicción ponderada se aplica dependiendo de si el fragmento es un fragmento de tipo B o un fragmento de tipo P, tal como se muestra en los bloques 1828 a 1830. En el ejemplo ilustrado, la tabla de ponderación predicha se lee si el fragmento es un fragmento de tipo P y se configura un indicador de predicción ponderado, o si el fragmento es un fragmento de tipo B y se configura un indicador de doble predicción ponderado. Esto se ilustra en la sintaxis 1712.

A partir de la cabecera, se determina un número máximo de candidatos de predictor del vector de movimiento temporal combinados para el fragmento, tal como se muestra en el bloque 1834 y en la sintaxis de la cabecera del fragmento 1714 de la figura 17A. Se debe observar, de nuevo, que el procesamiento que se muestra en la sintaxis

1714 se puede llevar a cabo sin referencia (por ejemplo, utilizando una lógica condicional) al tipo de fragmento, ya que esta operación es pertinente para todos los fragmentos de tipo P y fragmentos de tipo B, y esta condición se determinó en la sintaxis 1701.

A continuación, se lee un valor inicial para calcular el factor de cuantificación para el fragmento (slice_qp_delta) de acuerdo con la cabecera del fragmento. Esto se ilustra en el bloque 1836 de la figura 18B y en la sintaxis 1716 de la figura 17A.

Los parámetros del filtro de desbloqueo se leen, tal como se muestra en el bloque 1838 de la figura 18B y en la sintaxis 1718 de la figura 17A. La sintaxis de desbloqueo 1718 no cambia con respecto a la sintaxis de control del filtro de desbloqueo de referencia ilustrada en la figura 16B. Del mismo modo, el procesamiento del filtro de bucle adaptativo que se muestra en las sintaxis 1720 y 1722 permanece sin cambios con respecto al procesamiento del filtro de bucle adaptativo que se muestra en la sintaxis 1640 a 1642 de la figura 16C. Finalmente, la sintaxis de procesamiento de recuadro 1724 también se modifica a la sintaxis de procesamiento de recuadro 1644 de la figura 16C.

Segunda solución: agregar parámetro de cuantificación de fragmento inicial
Valores y parámetros de desbloqueo después de las condiciones lógicas de tipo P y B agrupadas

Esta solución es similar a la primera solución, en que la sintaxis correspondiente y las llamadas a funciones llevadas a cabo bajo la condición de que el tipo de fragmento es P o B son agrupadas y colocadas después del slice_qp_delta y la sintaxis relacionada con el desbloqueo.

Las figuras 19A y 19B son diagramas que ilustran la sintaxis de la cabecera del fragmento modificada de la segunda solución. Con referencia en primer lugar a la figura 19A, la sintaxis 1716 de slice_qp_data y la sintaxis 1718 relacionada con el desbloqueo se ejecuta antes de la sintaxis de la instrucción condicional que prueba si el fragmento es un fragmento inter-predicho. En la sintaxis 1701, esto se lleva a cabo determinando si el fragmento es de tipo P o de tipo B (P | | B), pero esto también se puede implementar determinando si el fragmento no es de tipo I. La sintaxis 1702 a 1712 (descrita anteriormente) se ejecuta si el fragmento es un fragmento de tipo P o un fragmento de tipo B, y la sintaxis 1720 a 1724 (pero no condicionada por la prueba realizada mediante la sintaxis 1701 para determinar si el fragmento es un fragmento P o un fragmento B.)

Tercera solución:

Agrupar las condiciones lógicas de tipo P y B en dos grupos
Esta solución también es similar a las soluciones primera y segunda, en que la sintaxis correspondiente y las llamadas a funciones llevadas a cabo bajo la condición que el tipo de fragmento es P o B son agrupadas. En esta solución, la posición de slice_qp_delta y la sintaxis relacionada con el desbloqueo en el CD actual permanecen sin cambios, pero la sintaxis relativa y las llamadas a funciones bajo la condición de tipo de fragmento igual a P / B antes y después de slice_qp_delta y la sintaxis relacionada con el desbloqueo son agrupadas por separado bajo instrucciones condicionales separadas.

Las figuras 20A y 20B son diagramas que ilustran la sintaxis de la cabecera del fragmento modificada de la tercera solución. Con referencia en primer lugar a la figura 19A, una instrucción condicional 1701 prueba si el fragmento es un fragmento inter-predicho. Una vez más, esto se puede realizar mediante una única instrucción lógica condicional si (slice_type == P | | slice_type == B) o si (slice_type!I). Si el fragmento es un fragmento inter-predicho, se ejecutan los elementos de sintaxis 1702 a 1708, incluida la sintaxis 1702 relacionada con habilitar los predictores de vectores de movimiento temporal, la sintaxis 1704 relacionada con la lógica de anulación de la imagen de referencia, la sintaxis 1705 relacionada con la modificación de la lista de imágenes de referencia, la sintaxis 1706 relacionada con la estructura de sintaxis de codificación de las diferencias del vector de movimiento (mvd_11_zero_flag) y la sintaxis 1708 relacionadas con la lógica de cabac.

Para todos los tipos de fragmento (incluido el tipo I), la cabecera de fragmento lee, a continuación, el parámetro slice_qp_delta y lleva a cabo las operaciones de desbloqueo que se muestran en las sintaxis 1716 y 1718.

Volviendo a la figura 20B, otra instrucción condicional para determinar si el tipo de fragmento es un fragmento inter-predicho. De nuevo, esto se puede conseguir con una única prueba de instrucción condicional para P | | B tal como se muestra en la sintaxis 2002, o con una única prueba de instrucción condicional para !I (no I). Si el enunciado condicional de 2002 resulta ser positivo, se ejecuta la sintaxis de la lógica del predictor del vector de movimiento temporal y la sintaxis 1712 relacionada con la tabla de predicción ponderada, y se lee el número máximo de candidatos de fusión, tal como se muestra en las sintaxis 1710, 1712 y 1714, respectivamente. La lógica de la cabecera de sintaxis restante 1720 a 1724 se ejecuta tal como se indicó anteriormente.

Entorno de hardware

La figura 20 ilustra un sistema de procesamiento 2000a modo de ejemplo, que se podría utilizar para implementar las realizaciones de la invención. El ordenador 2002 comprende un procesador 2004 y una memoria, tal como una memoria de acceso aleatorio (RAM – Read Only Memory, en inglés) 2006. El ordenador 2002 está acoplado de manera operativa a una pantalla 2022, que presenta imágenes como ventanas al usuario en una interfaz gráfica de usuario 2018B. el ordenador 2002 puede estar acoplado a otros dispositivos, tales como un teclado 2014, un dispositivo de ratón 2016, una impresora, etc. Por supuesto, los expertos en la materia reconocerán que cualquier combinación de los componentes anteriores, o cualquier número de componentes, periféricos, y otros dispositivos diferentes pueden ser utilizados con el ordenador 2002.

En general, el ordenador 2002 funciona bajo el control de un sistema operativo 2008 almacenado en la memoria 2006, e interactúa con el usuario para aceptar entradas y comandos y presentar resultados a través de un módulo de interfaz gráfica de usuario (GUI – Graphical User Interface, en inglés) 2018A. Aunque el módulo de GUI 2018A se representa como un módulo separado, las instrucciones que llevan a cabo las funciones de la GUI pueden residir o estar distribuidas en el sistema operativo 2008, el programa informático 2010 o estar implementadas con una memoria y procesadores de propósito especial. El ordenador 2002 implementa, asimismo, un compilador 2012 que permite que un programa informático 2010 escrito en un lenguaje de programación tal como COBOL, C++, FORTRAN u otro lenguaje sea traducido al código legible por el procesador 2004. Una vez completada, la aplicación 2010 accede y manipula los datos almacenados en la memoria 2006 del ordenador 2002 utilizando las relaciones y la lógica que se generó utilizando el compilador 2012. El ordenador 2002 comprende, asimismo, opcionalmente, un dispositivo de comunicación externo, tal como un modem, un enlace satelital, una tarjeta Ethernet u otro dispositivo para comunicarse con otros ordenadores.

En una realización, las instrucciones que implementan el sistema operativo 2008, el programa informático 2010 y el compilador 2012 están materialmente incorporadas en un medio legible por un ordenador, por ejemplo, un dispositivo de almacenamiento de datos 2020, que podría incluir uno o varios dispositivos de almacenamiento de datos fijos o extraíbles, tales como una unidad de compresión, una unidad de disquete 2024, un disco duro, una unidad de CD-ROM, unidad de cinta, etc. Además, el sistema operativo 2008 y el programa informático 2010 están compuestos de instrucciones que, cuando son leídas y ejecutadas por el ordenador 2002, hacen que el ordenador 2002 lleve a cabo las etapas necesarias para implementar y/o utilizar la invención. El programa informático 2010 y/o las instrucciones de funcionamiento también pueden estar incorporadas de manera tangible en la memoria 2006 y/o en los dispositivos de comunicación de datos 2030, lo que hace un producto de programa informático o artículo de fabricación. De este modo, los términos “artículo de fabricación”, “dispositivo de almacenamiento de programas” y “producto de programa informático”, tal como se utilizan en el presente documento, pretenden abarcar un programa informático accesible desde cualquier dispositivo o medio legible por un ordenador.

El sistema de procesamiento 2000 también puede estar incrustado en un ordenador de sobremesa, un ordenador portátil, una tableta, un ordenador de libreta de notas, asistente de datos personal (PDA – Personal Data Assistant, en inglés), un teléfono celular, un teléfono inteligente o cualquier dispositivo con capacidad de procesamiento y memoria adecuados. Además, el sistema de procesamiento 2000 puede utilizar hardware de propósito especial para realizar algunas o todas las funcionalidades indicadas anteriormente. Por ejemplo, los procesos de codificación y descodificación descritos anteriormente pueden ser llevados a cabo por un procesador de propósito especial y la memoria asociada.

Los expertos en la técnica reconocerán que se pueden realizar muchas modificaciones a esta configuración sin apartarse del alcance de la presente invención. Por ejemplo, los expertos en la materia reconocerán que se puede utilizar cualquier combinación de los componentes anteriores, o cualquier número de componentes, periféricos y otros dispositivos diferentes. Por ejemplo, las funciones particulares descritas en el presente documento pueden ser realizadas mediante módulos de hardware o un procesador que ejecuta instrucciones almacenadas en forma de software o firmware. Además, la funcionalidad descrita en el presente documento puede ser combinada en módulos individuales o expandida para ser realizada en múltiples módulos.

Conclusión

La descripción anterior de la realización preferida se ha presentado con fines de ilustración y de descripción. No pretende ser exhaustiva o limitar la invención a la forma precisa descrita. Muchas modificaciones y variaciones son posibles a la luz de la explicación anterior. Se pretende que el alcance de los derechos no esté limitado por esta descripción detallada, sino más bien por las reivindicaciones adjuntas a la misma.

REIVINDICACIONES

1. Un método para descodificar una secuencia que comprende una pluralidad de imágenes, pudiendo estar dividida cada una de la pluralidad de imágenes en uno o varios fragmentos, estando procesado cada uno de los uno o varios fragmentos, al menos en parte, de acuerdo con una cabecera de fragmento, comprendiendo el método:

determinar, utilizando una instrucción condicional en la prueba de tipo de fragmento de los datos del tipo de fragmento, si un fragmento de los uno o varios fragmentos es un fragmento inter-predicho de acuerdo con los datos del tipo de fragmento probando la única instrucción condicional, si el fragmento no es un fragmento de tipo I de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento, en el que:

el fragmento es un fragmento de tipo B (1504) si la al menos una unidad de codificación del fragmento está codificada utilizando la compensación de movimiento de acuerdo con hasta dos vectores de movimiento asociados con al menos una imagen de referencia;
el fragmento es un fragmento de tipo P (1504) si al menos una unidad de codificación del fragmento está codificada utilizando la compensación de movimiento de acuerdo con no más de un vector de movimiento asociado con la al menos una imagen de referencia;
el fragmento es un fragmento de tipo I (1502) si ninguna unidad de codificación del fragmento está codificada utilizando la compensación de movimiento; y
el fragmento es un fragmento inter-predicho solo si es un fragmento de tipo P o un fragmento de tipo B;
si se determina que el fragmento es un fragmento inter-predicho (1802), ejecutar, con una prueba de tipo de fragmento adicional realizada solo para identificar el fragmento inter-predicho como un fragmento de tipo P o un fragmento de tipo B, una lógica adicional dentro de una instrucción condicional en el siguiente orden:

(a) determinar (1804) si un indicador del predictor del vector de movimiento temporal de secuencia indica que la secuencia en la que se muestra la imagen puede estar codificada utilizando el, al menos, un predictor de vector de movimiento temporal;
(b) si el indicador del predictor del vector de movimiento temporal de secuencia indica que la secuencia en la que está dispuesta la imagen puede estar codificada utilizando el, al menos, un predictor de vector de movimiento temporal, leer (1806) un indicador de habilitación del predictor del vector de movimiento temporal del fragmento / imagen que indica que la imagen en la que está dispuesto el fragmento está codificada utilizando, al menos, un predictor de vector de movimiento temporal;
(c) leer (1808) un indicador de anulación de imagen de referencia que indica si los datos que describen un índice máximo de, al menos, una lista de imágenes de referencia que tienen, al menos, una imagen de referencia, están incluidos en la cabecera del fragmento;
(d) si el indicador de anulación de imagen de referencia leído indica que los datos que describen el índice máximo de la, al menos una, lista de imágenes de referencia que tienen, al menos, una imagen de referencia, están presentes (1810), leer (1812) los datos;
(e) leer (1816) la información de modificación de la lista de imágenes de referencia solo si la, al menos, una imagen de referencia está explícitamente definida (1814);
(f) leer (1820) la información que indica una estructura de diferencias de vectores de movimiento utilizada en la compensación de movimiento si el fragmento es un fragmento B (1818);
(g) leer (1822) la información que define la inicialización de las variables de contexto para la codificación por entropía del fragmento solo si la información que define la inicialización de las variables de contexto está presente en la cabecera de fragmento;
(h) determinar (1824) si el fragmento está codificado utilizando, al menos, un predictor de vector de movimiento temporal de acuerdo con el indicador de habilitación del predictor de vector de movimiento temporal del fragmento / imagen leído;
(i) solo si el fragmento está codificado utilizando, al menos, un predictor de vector de movimiento temporal, leer (1826) la información que describe, al menos, una imagen de referencia del fragmento asociada con, al menos, un predictor de vector de movimiento temporal, al menos en parte de acuerdo con si el fragmento es un fragmento de tipo P o un fragmento de tipo B;
(j) si la predicción ponderada está habilitada para el fragmento, leer (1832) una tabla de ponderación de predicción; y
(k) determinar (1834), a partir de la cabecera del fragmento, un número máximo de candidatos para el, al menos un predictor de vector de movimiento temporal para el fragmento; en el que ninguna de las etapas (a) - (k) son llevadas a cabo para un fragmento que no se determina que es un fragmento inter-predicho.

2. El método de la reivindicación 1, en el que determinar si un fragmento de los uno o varios fragmentos es un fragmento inter-predicho de acuerdo con los datos del tipo de fragmento comprende:

probar, en una instrucción condicional, si el fragmento es un fragmento de tipo P o un fragmento de tipo B de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento.

3. El método de la reivindicación 1, que comprende, además:

leer una variable que describe un valor inicial para un parámetro de cuantificación después de que se complete la lógica adicional en la etapa (k) si la instrucción condicional determina que el fragmento es un fragmento inter-predicho; o
leer la variable que describe el valor inicial para el parámetro de cuantificación sin ejecutar la lógica adicional de las etapas (a) - (k) si la única instrucción condicional determina que el fragmento no es un fragmento inter-predicho.

4. Un método para codificar una secuencia que comprende una pluralidad de imágenes, pudiendo estar dividida cada una de la pluralidad de imágenes en uno o varios fragmentos, estando procesado cada uno de los uno o varios fragmentos, al menos en parte, de acuerdo con una cabecera de fragmento, comprendiendo el método:

codificar los datos del tipo de fragmento para un fragmento de los uno o varios fragmentos en la cabecera, indicando los datos del tipo de fragmento el tipo de fragmento, en el que:

el fragmento es un fragmento de tipo B (1504) si la al menos una unidad de codificación del fragmento está codificada utilizando la compensación de movimiento de acuerdo con hasta dos vectores de movimiento asociados con al menos una imagen de referencia;
el fragmento es un fragmento de tipo P (1504) si al menos una unidad de codificación del fragmento está codificada utilizando la compensación de movimiento de acuerdo con no más de un vector de movimiento; y
el fragmento es un fragmento de tipo I (1502) si ninguna unidad de codificación del fragmento está codificada utilizando la compensación de movimiento;
el fragmento es un fragmento inter-predicho si es un fragmento de tipo P o un fragmento de tipo B;
generar una lógica de cabecera de fragmento para determinar, utilizando una instrucción condicional en la prueba de tipo de fragmento de los datos del tipo de fragmento, si un fragmento de los uno o varios fragmentos es un fragmento inter-predicho de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento probando la única instrucción condicional, si el fragmento no es un fragmento de tipo I de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento;
generar una lógica de cabecera de fragmento que es lógica adicional dentro de una instrucción condicional para llevar a cabo, en el siguiente orden y con pruebas adicionales de tipo de fragmento llevadas a cabo solo para identificar el fragmento inter-predicho como un fragmento de tipo P o un fragmento de tipo B, operaciones que comprenden las operaciones (a) - (k) de si el fragmento se determinó que es un fragmento inter-predicho (1802):

(a) determinar (1804) si un indicador del predictor del vector de movimiento temporal de secuencia indica que la secuencia en la que está dispuesta la imagen puede estar codificada utilizando el, al menos, un predictor de vector de movimiento temporal;
(b) si el indicador del predictor del vector de movimiento temporal de secuencia indica que la secuencia en la que está dispuesta la imagen puede estar codificada utilizando el, al menos, un predictor de vector de movimiento temporal, leer (1806) un indicador de habilitación del predictor del vector de movimiento temporal del fragmento / imagen que indica que la imagen en la que está dispuesto el fragmento está codificada utilizando, al menos, un predictor de vector de movimiento temporal;
(c) leer (1808) un indicador de anulación de imagen de referencia que indica si los datos que describen un índice máximo de, al menos, una lista de imágenes de referencia que tienen, al menos, una imagen de referencia, están incluidos en la cabecera del fragmento;
(d) si el indicador de anulación de imagen de referencia leído indica que los datos que describen el índice máximo de la, al menos una, lista de imágenes de referencia que tienen, al menos, una imagen de referencia, están presentes (1810), leer (1812) los datos;
(e) leer (1816) la información de modificación de la lista de imágenes de referencia solo si la, al menos, una imagen de referencia está explícitamente definida (1814);
(f) leer (1820) la información que indica una estructura de diferencias de vectores de movimiento utilizada en la compensación de movimiento si el fragmento es un fragmento B (1818);
(g) leer (1822) la información que define la inicialización de las variables de contexto para la codificación por entropía del fragmento solo si la información que define la inicialización de las variables de contexto está presente en la cabecera de fragmento;
(h) determinar (1824) si el fragmento está codificado utilizando, al menos, un predictor de vector de movimiento temporal de acuerdo con el indicador de habilitación del predictor de vector de movimiento temporal del fragmento / imagen leído;

(i) solo si el fragmento está codificado utilizando, al menos, un predictor de vector de movimiento temporal, leer (1826) la información que describe, al menos, una imagen de referencia del fragmento asociada con, al menos, un predictor de vector de movimiento temporal, al menos en parte de acuerdo con si el fragmento es un fragmento de tipo P o un fragmento de tipo B;
(j) si la predicción ponderada está habilitada para el fragmento, leer (1832) una tabla de ponderación de predicción; y
(l) determinar (1834), a partir de la cabecera del fragmento, un número máximo de candidatos para el, al menos un predictor de vector de movimiento temporal para el fragmento; en el que

ninguna de las etapas (a) - (k) son llevadas a cabo para un fragmento que no se determina que es un fragmento inter-predicho.

5. El método de la reivindicación 4, en el que determinar si un fragmento de los uno o varios fragmentos es un fragmento inter-predicho de acuerdo con los datos del tipo de fragmento, comprende:

probar, en una instrucción condicional, si el fragmento es un fragmento de tipo P o un fragmento de tipo B de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento.

6. El método de la reivindicación 4, que comprende, además, generar una lógica de cabecera de fragmento para:

leer una variable que describe un valor inicial para un parámetro de cuantificación después de que se complete la lógica adicional en la etapa (k) si la única instrucción condicional determina que el fragmento es un fragmento inter-predicho; o
leer la variable que describe el valor inicial para el parámetro de cuantificación sin ejecutar la lógica adicional de las etapas (a) - (k) si la única instrucción condicional determina que el fragmento es un fragmento de tipo I.

7. Un dispositivo de procesamiento (2100), para descodificar una secuencia que comprende una pluralidad de imágenes, pudiendo estar dividida cada una de la pluralidad de imágenes en uno o varios fragmentos, estando destinado cada uno de los fragmentos a ser procesados, al menos en parte, de acuerdo con una cabecera de fragmento, estando provisto el dispositivo de procesamiento (2100) de un procesador (21 04A, 21 04B) para ejecutar las instrucciones almacenadas en una memoria (2106) acoplada de manera comunicativa mediante la cual el dispositivo de procesamiento (2100) está operativo para:

determinar, utilizando una instrucción condicional en la prueba de tipo de fragmento de los datos del tipo de fragmento, si un fragmento de los uno o varios fragmentos es un fragmento inter-predicho de acuerdo con los datos del tipo de fragmento probando la única instrucción condicional, si el fragmento no es un fragmento de tipo I de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento, en el que:

el fragmento es un fragmento de tipo B (1504) si la al menos una unidad de codificación del fragmento está codificada utilizando la compensación de movimiento de acuerdo con hasta dos vectores de movimiento asociados con al menos una imagen de referencia;
el fragmento es un fragmento de tipo P (1504) si al menos una unidad de codificación del fragmento está codificada utilizando la compensación de movimiento de acuerdo con no más de un vector de movimiento asociado con la al menos una imagen de referencia;
el fragmento es un fragmento de tipo I (1502) si ninguna unidad de codificación del fragmento está codificada utilizando la compensación de movimiento; y
el fragmento es un fragmento inter-predicho solo si es un fragmento de tipo P o un fragmento de tipo B;
si se determina que el fragmento es un fragmento inter-predicho (1802), ejecutar, con una prueba de tipo de fragmento adicional realizada solo para identificar el fragmento inter-predicho como un fragmento de tipo P o un fragmento de tipo B, una lógica adicional dentro de una instrucción condicional en el siguiente orden:

(a) determinar (1804) si un indicador del predictor del vector de movimiento temporal de secuencia indica que la secuencia en la que está dispuesta la imagen puede estar codificada utilizando el, al menos, un predictor de vector de movimiento temporal;
(b) si el indicador del predictor del vector de movimiento temporal de secuencia indica que la secuencia en la que está dispuesta la imagen puede estar codificada utilizando el, al menos, un predictor de vector de movimiento temporal, leer (1806) un indicador de habilitación del predictor del vector de movimiento temporal del fragmento / imagen que indica que la imagen en la que está dispuesto el fragmento está codificada utilizando, al menos, un predictor de vector de movimiento temporal;
(c) leer (1808) un indicador de anulación de imagen de referencia que indica si los datos que describen un índice máximo de, al menos, una lista de imágenes de referencia que tienen, al menos, una imagen de referencia, están incluidos en la cabecera del fragmento;

(d) si el indicador de anulación de imagen de referencia leído indica que los datos que describen el índice máximo de la, al menos una, lista de imágenes de referencia que tienen, al menos, una imagen de referencia, están presentes (1810), leer (1812) los datos;

(e) leer (1816) la información de modificación de la lista de imágenes de referencia solo si la, al menos, una imagen de referencia está explícitamente definida (1814);

(f) leer (1820) la información que indica una estructura de diferencias de vectores de movimiento utilizada en la compensación de movimiento si el fragmento es un fragmento B (1818);

(g) leer (1822) la información que define la inicialización de las variables de contexto para la codificación por entropía del fragmento solo si la información que define la inicialización de las variables de contexto está presente en la cabecera de fragmento;

(h) determinar (1824) si el fragmento está codificado utilizando, al menos, un predictor de vector de movimiento temporal de acuerdo con el indicador de habilitación del predictor de vector de movimiento temporal del fragmento / imagen leído;

(i) solo si el fragmento está codificado utilizando, al menos, un predictor de vector de movimiento temporal, leer (1826) la información que describe, al menos, una imagen de referencia del fragmento asociada con, al menos, un predictor de vector de movimiento temporal, al menos en parte de acuerdo con si el fragmento es un fragmento de tipo P o un fragmento de tipo B;

(j) aplicar (1832) una predicción ponderada de acuerdo con si el fragmento es un fragmento de tipo B o un fragmento de tipo P; y

(k) determinar (1834), a partir de la cabecera del fragmento, un número máximo de candidatos de predictor de vector de movimiento temporal fusionados para el fragmento; en el que

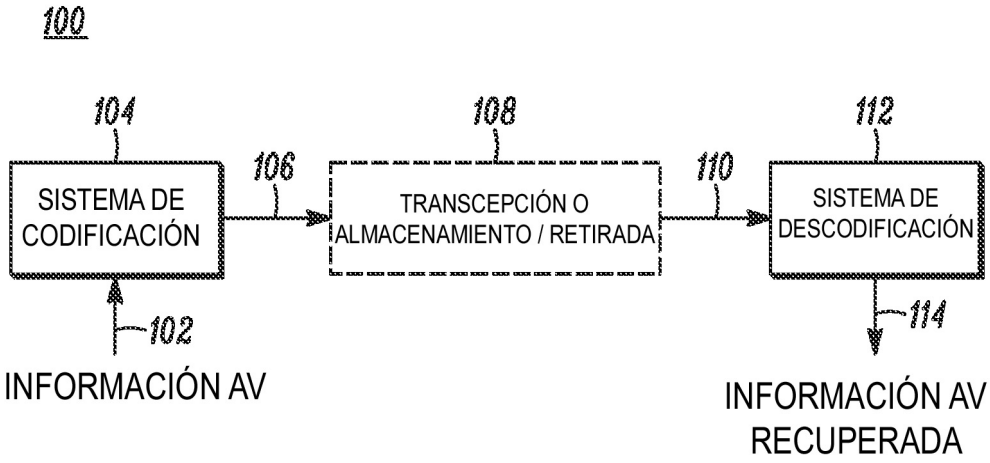
ninguna de las etapas (a) - (k) son llevadas a cabo para un fragmento que no se determina que es un fragmento inter-predicho.

8. El dispositivo de procesamiento de la reivindicación 7 en el que el dispositivo de procesamiento (2100) puede funcionar, adicionalmente, para determinar si un fragmento de los uno o varios fragmentos es un fragmento inter-predicho de acuerdo con los datos del tipo de fragmento probando, en la única instrucción condicional, si el fragmento es un fragmento de tipo P o un fragmento de tipo B de acuerdo con los datos del tipo de fragmento leídos de la cabecera del fragmento.

9. El dispositivo de procesamiento de la reivindicación 7, que comprende, además, instrucciones para:

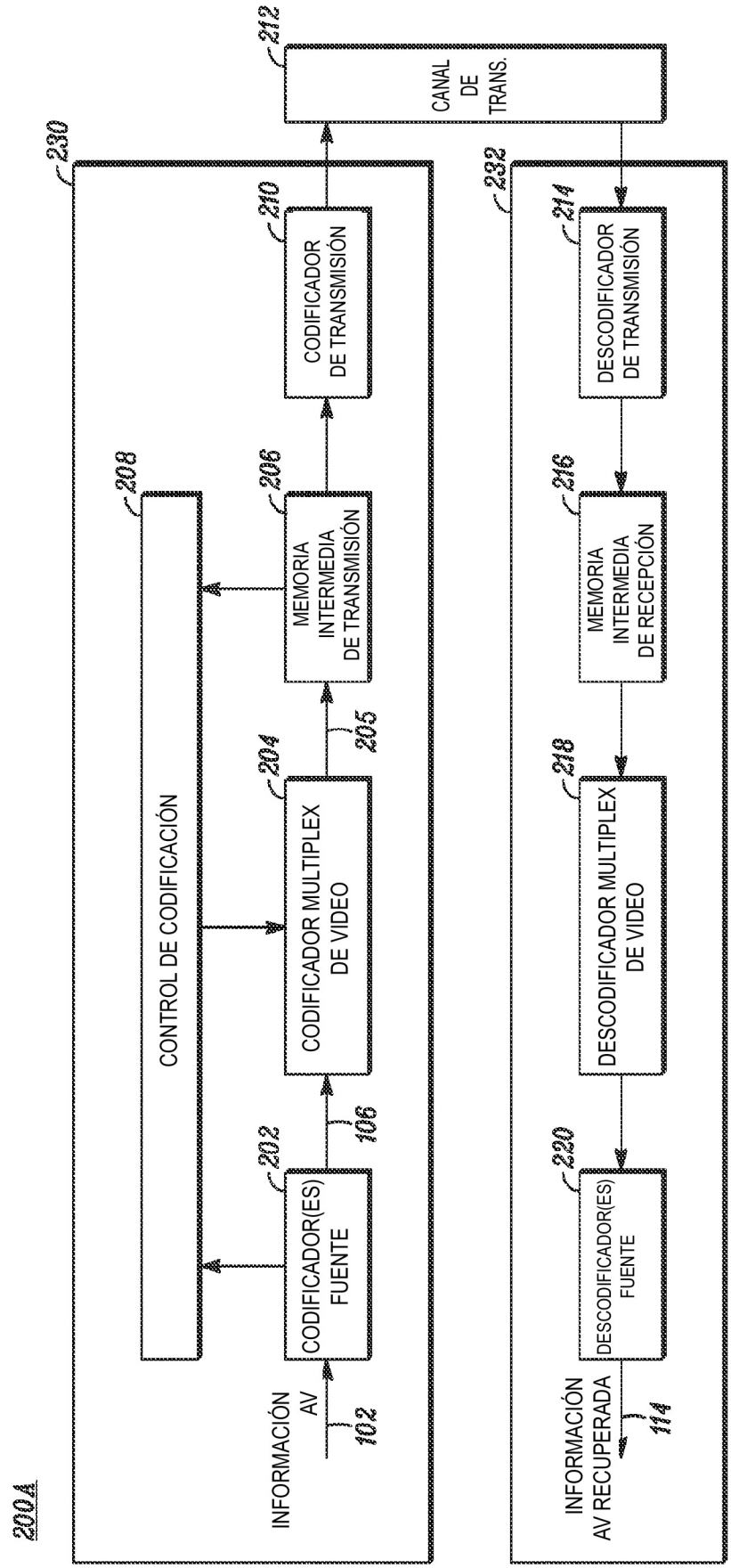
leer una variable que describe un valor inicial para un parámetro de cuantificación después de que se ha completado la lógica adicional en la etapa (k) si la instrucción condicional determina que el fragmento es un fragmento inter-predicho; o

leer la variable que describe el valor inicial para el parámetro de cuantificación sin ejecutar la lógica adicional de las etapas (a) - (k) si la única instrucción condicional determina que el fragmento no es un fragmento predicho.



TÉCNICA ANTERIOR

FIG. 1



TÉCNICA ANTERIOR

FIG. 2A

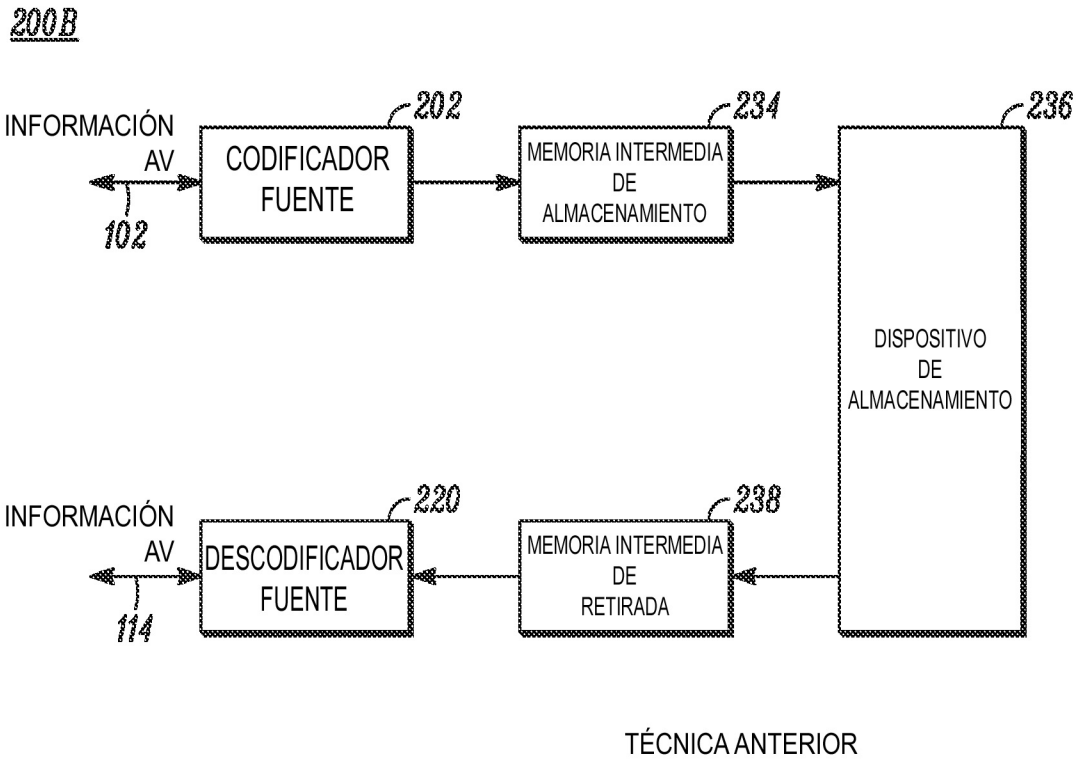

200B
INFORMACIÓN AV
102
CODIFICADOR FUENTE
202
MEMORIA INTERMEDIA DE ALMACENAMIENTO
234
DISPOSITIVO DE ALMACENAMIENTO
236
INFORMACIÓN AV
114
DESCODIFICADOR FUENTE
220
MEMORIA INTERMEDIA DE RETIRADA
238
TÉCNICA ANTERIOR

FIG. 2B

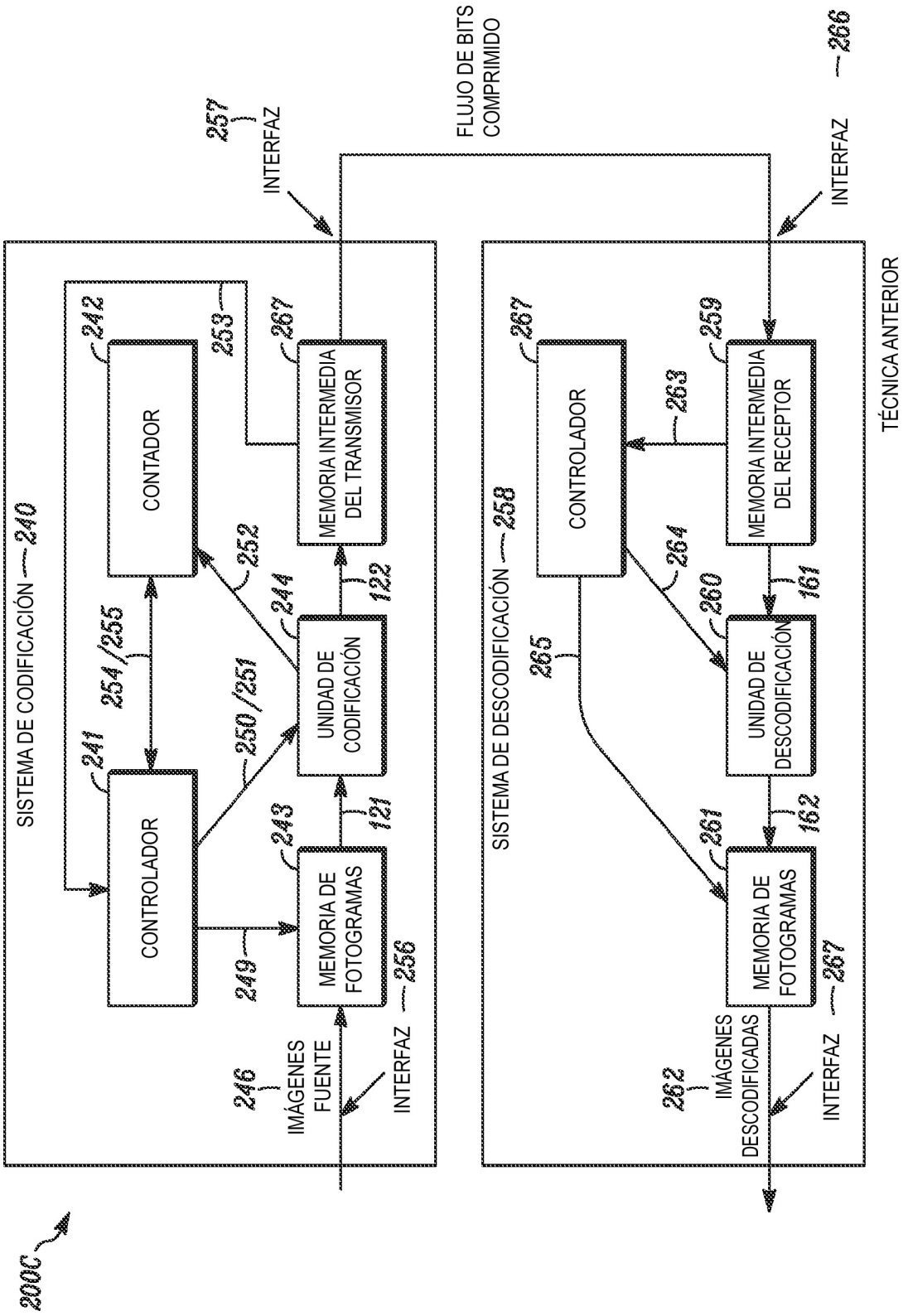

200C
SISTEMA DE CODIFICACIÓN 240
SISTEMA DE DESCODIFICACIÓN 258
CONTROLADOR
CONTADOR
MEMORIA DE FOTOGRAMAS
UNIDAD DE CODIFICACIÓN
MEMORIA INTERMEDIA DEL TRANSMISOR
MEMORIA INTERMEDIA DEL RECEPTOR
UNIDAD DE DESCODIFICACIÓN
INTERFAZ
IMÁGENES FUENTE
IMÁGENES DESCODIFICADAS
FLUJO DE BITS COMPRIMIDO
TÉCNICA ANTERIOR
241
242
243
244
246
249
250/251
252
253
254/255
256
257
260
261
262
263
264
265
266
267
121
122
161
162

FIG. 2C

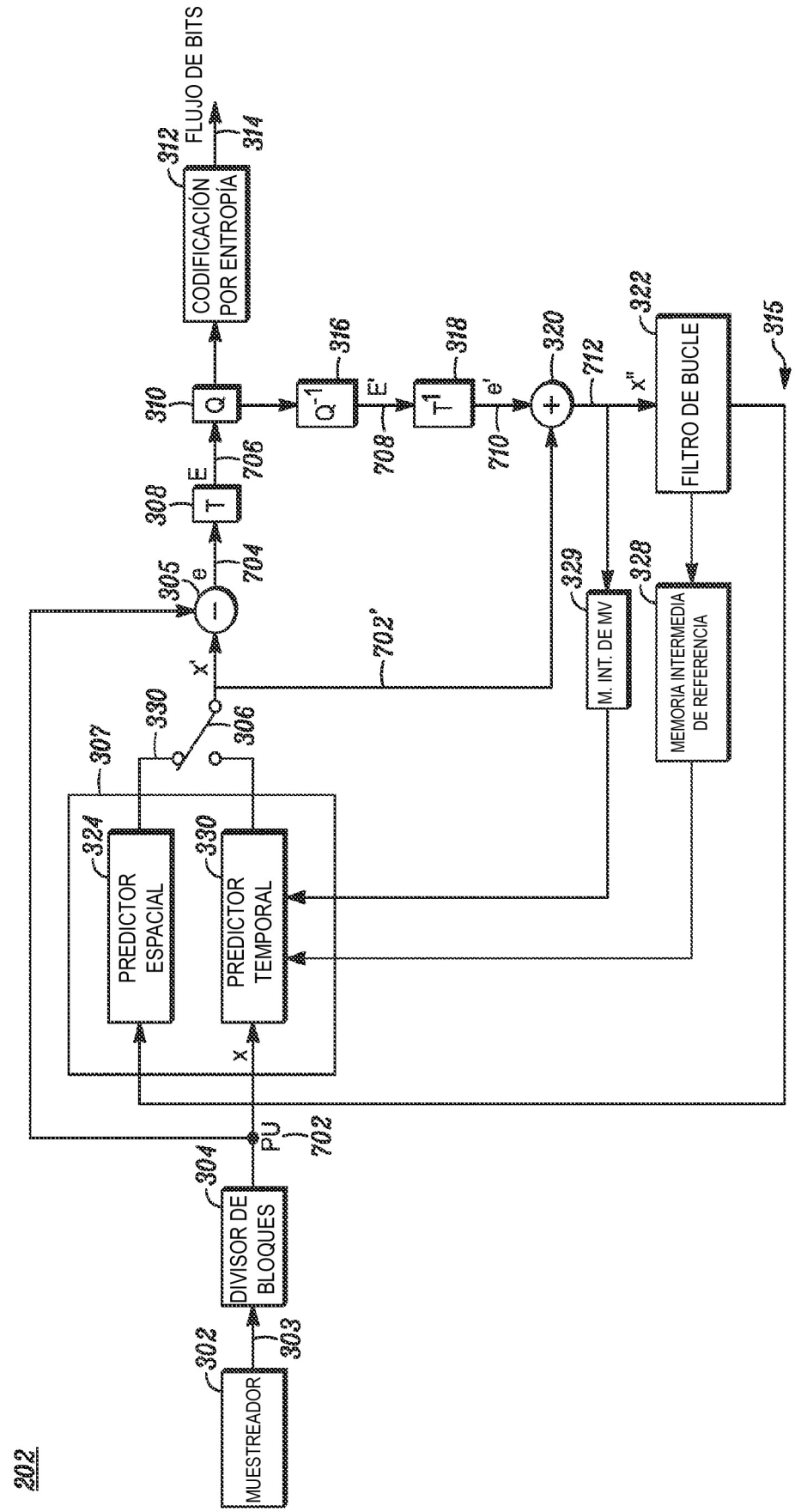

202
MUESTREADOR
302
303
DIVISOR DE BLOQUES
304
PU
702
307
PREDICTOR ESPACIAL
324
PREDICTOR TEMPORAL
330
330
306
x
x'
305
e
704
702'
T
E
308
706
Q
310
CODIFICACIÓN POR ENTROPÍA
312
FLUJO DE BITS
314
Q^{-1}
316
E'
708
T^{-1}
318
e'
710
320
712
x''
FILTRO DE BUCLE
322
315
M. INT. DE MV
329
MEMORIA INTERMEDIA DE REFERENCIA
328

FIG. 3

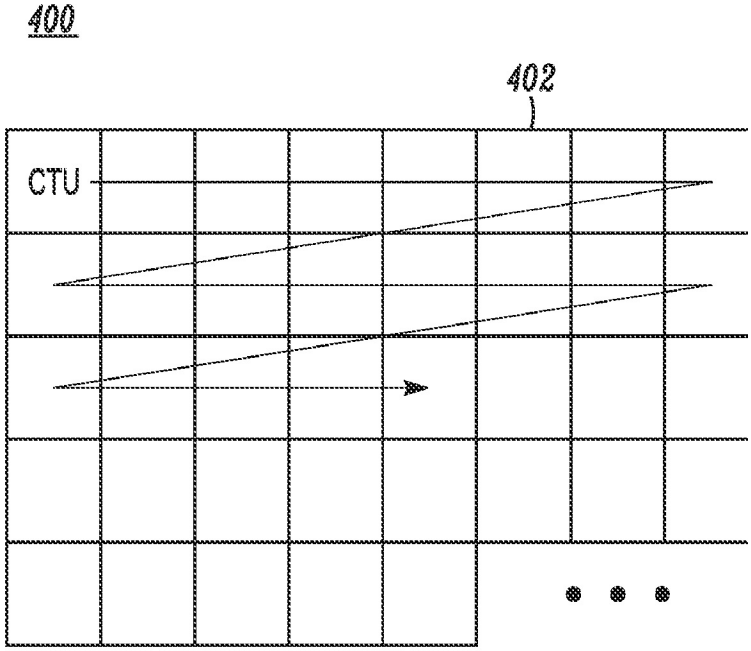


FIG. 4

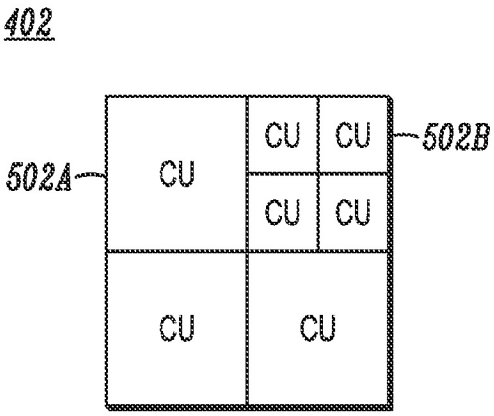


FIG. 5

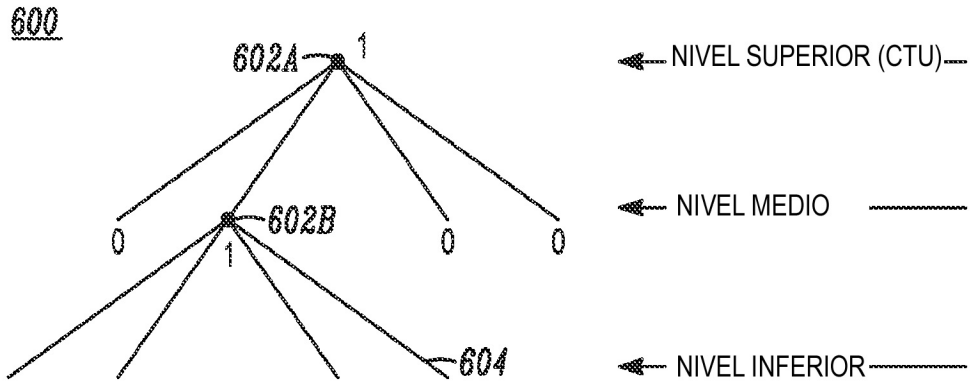

600
602A
1
NIVEL SUPERIOR (CTU)
NIVEL MEDIO
602B
0
1
0
0
604
NIVEL INFERIOR

FIG. 6

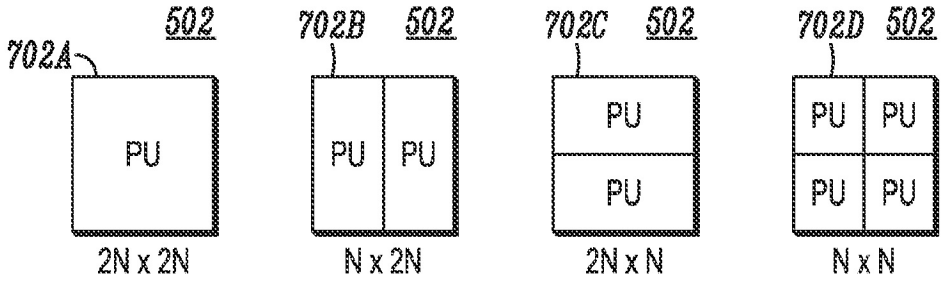

502
702A
PU
2N x 2N
702B
502
PU
PU
N x 2N
702C
502
PU
PU
2N x N
702D
502
PU
PU
PU
PU
N x N

FIG. 7

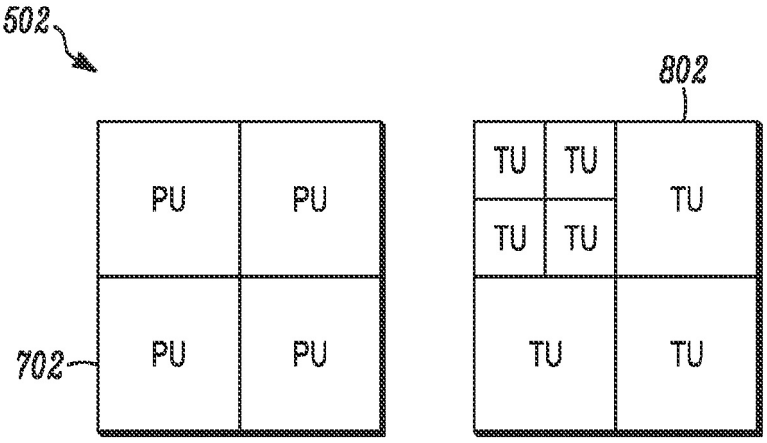

502
802
PU
PU
PU
PU
702
TU
TU
TU
TU
TU
TU
TU

FIG. 8

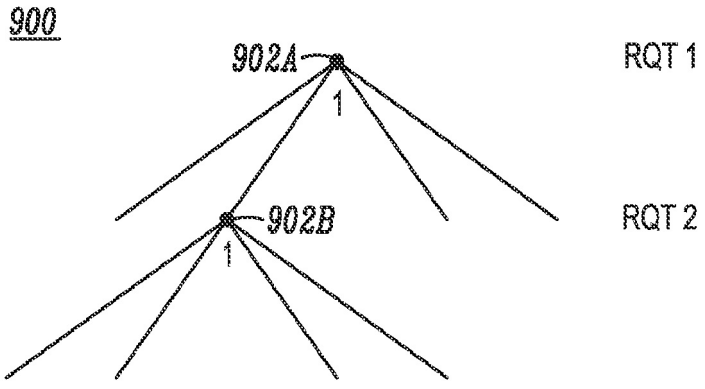

900
902A
1
RQT 1
902B
1
RQT 2

FIG. 9

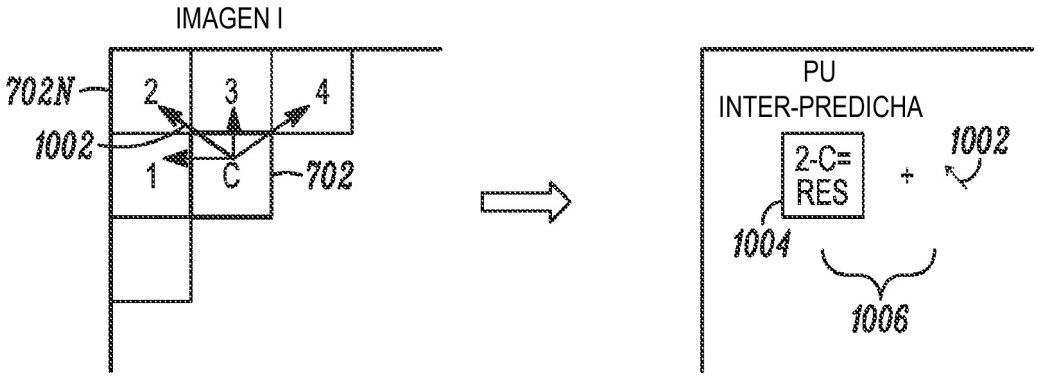


IMAGEN I
702N
2
3
4
1002
1
C
702
PU
INTER-PREDICHA
2-C=
RES
+
1002
1004
1006

FIG. 10

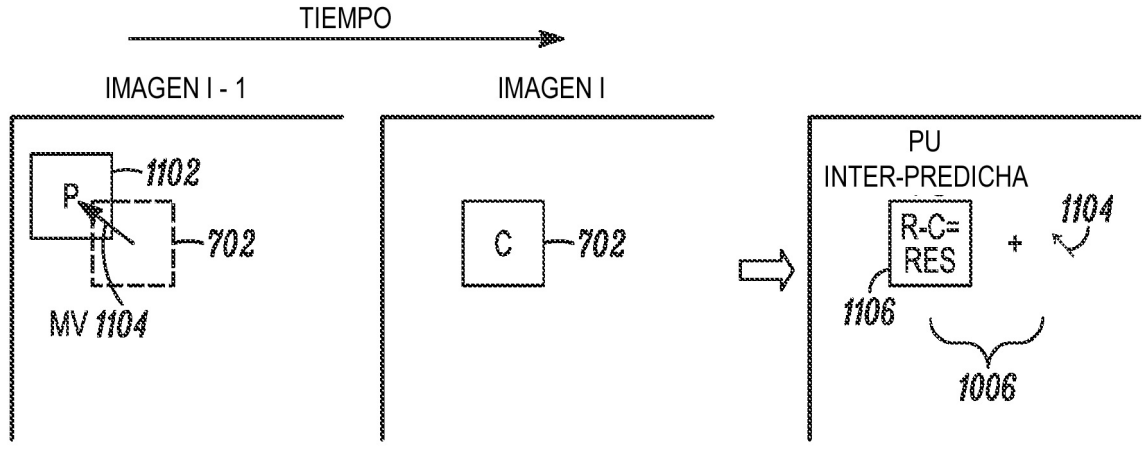

TIEMPO
IMAGEN I - 1
IMAGEN I
1102
P
702
MV 1104
C
702
PU
INTER-PREDICHA
R-C=
RES
+
1104
1106
1006

FIG. 11

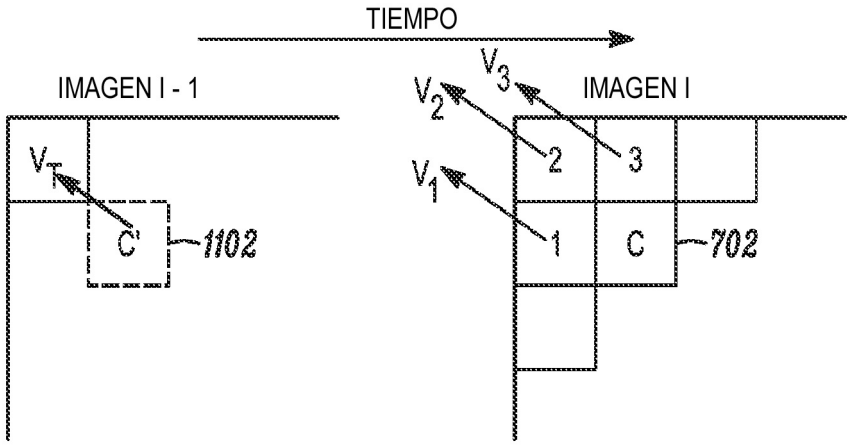


FIG. 12

P = LISTA 0

IDX	R PIX
0	4
1	2
2	0
3	6
4	8
5	10

P, B = LISTA 1

IDX	R PIX
0	6
1	8
2	10
3	4
4	2
5	0

FIG. 13

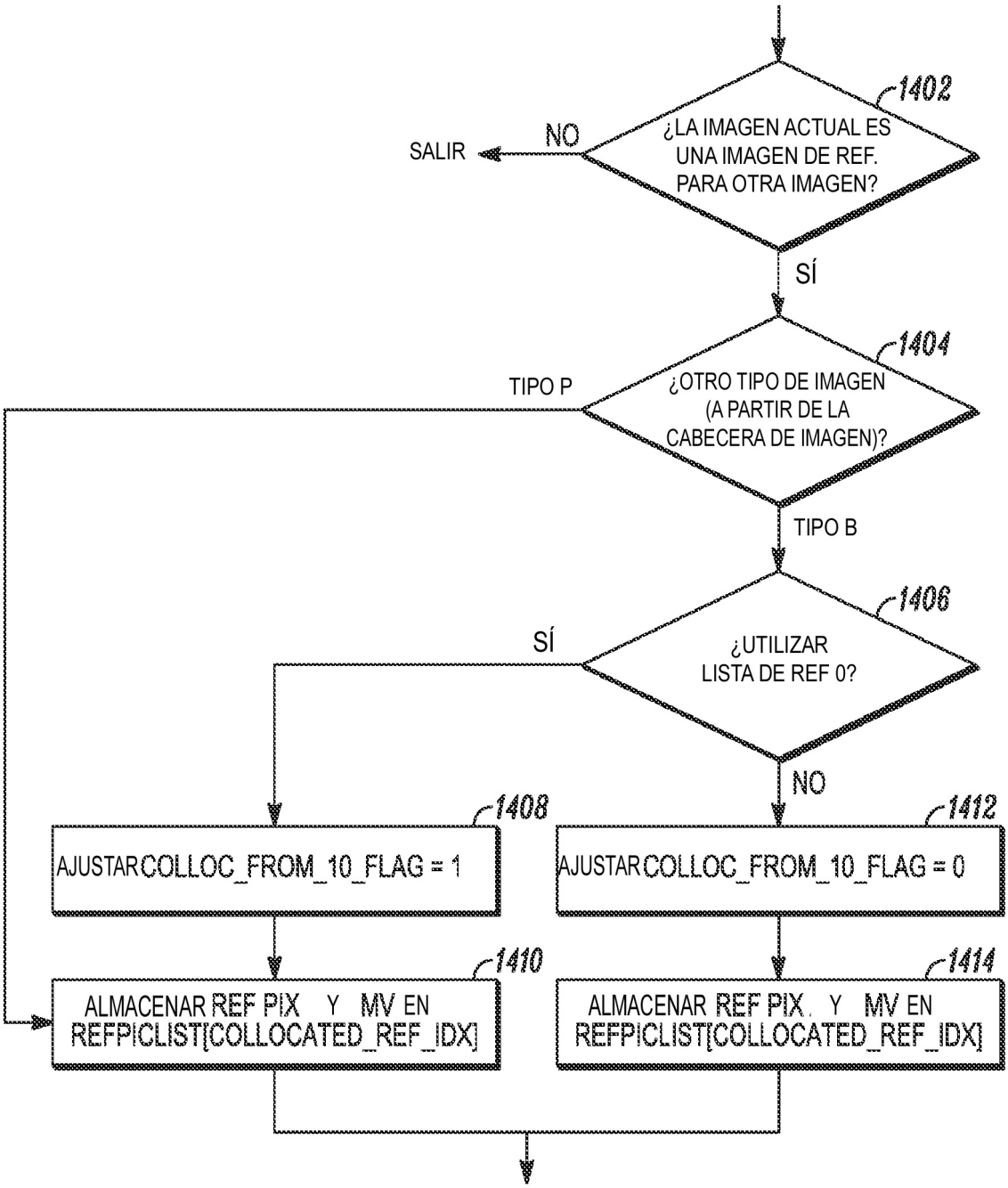

1402
¿LA IMAGEN ACTUAL ES UNA IMAGEN DE REF. PARA OTRA IMAGEN?
NO
SALIR
SÍ
1404
¿OTRO TIPO DE IMAGEN (A PARTIR DE LA CABECERA DE IMAGEN)?
TIPO P
TIPO B
1406
¿UTILIZAR LISTA DE REF 0?
SÍ
NO
1408
AJUSTAR COLLOC_FROM_10_FLAG = 1
1412
AJUSTAR COLLOC_FROM_10_FLAG = 0
1410
ALMACENAR REF PIX Y MV EN REFPICLIST[COLLOCATED_REF_IDX]
1414
ALMACENAR REF PIX Y MV EN REFPICLIST[COLLOCATED_REF_IDX]

FIG. 14

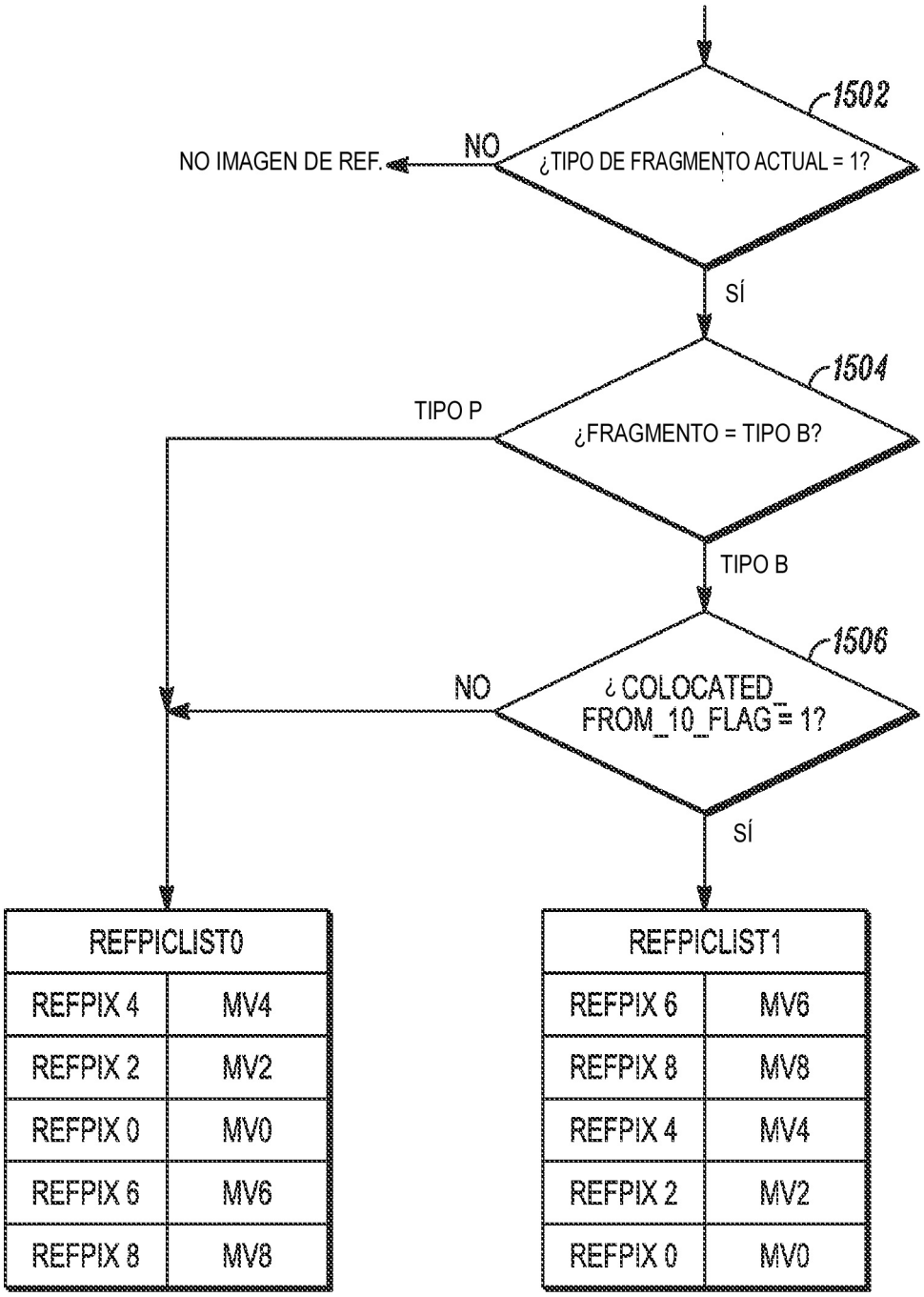

1502
¿TIPO DE FRAGMENTO ACTUAL = 1?
NO
NO IMAGEN DE REF.
SÍ
1504
¿FRAGMENTO = TIPO B?
TIPO P
TIPO B
1506
¿ COLOCATED_FROM_10_FLAG = 1?
NO
SÍ
REFPICLIST0
REFPIX 4
MV4
REFPIX 2
MV2
REFPIX 0
MV0
REFPIX 6
MV6
REFPIX 8
MV8
REFPICLIST1
REFPIX 6
MV6
REFPIX 8
MV8
REFPIX 4
MV4
REFPIX 2
MV2
REFPIX 0
MV0

FIG. 15

	slice_header() {	DESCRIPTOR
1602	first slice_in_pic_flag	u(l)
1604	pic_parameter_set_id	ue(v)
	if(!first_slice_in_pic_flag)	
1606	slice_address	u(v)
	if(dependent_slice_enabled_flag && !first & _slice_in_pic_flag)	
1608	dependent_slice flag	u(l)
	if(!dependent_slice_flag) {	
1610	slice_type	ue(v)
	if(output_flag_present_flag)	
	pic_output_flag	u(l)
	if(separate_colour_plane_flag == 1)	
	colour_plane_id	u(2)
	if(RapPicFlag) {	
	rap_pic_id	ue(v)
	no_output_of_prior_pics_flag	u(l)
	}	
	if(!IdrPicFlag) {	
	pic_order_cnt_lsb	u(v)
	short_term_ref_pic_set_sps_flat	u(l)
	if(!short_term_ref_pic_set_sps_flag)	
	short_term_ref_pic_set(num_short_term_ref_pic_sets)	
	Else	
	short_term_ref_pic_set_idx	u(v)
	if(long_term_ref_pics_present_flag) {	
1612	num_long_term_pics	ue(v)
	for(i=0; i< num_long_term_pics; i++) {	
	poc_lsb_lt[i]	u(v)
	delta_poc_msb_present_flag[i]	u(l)
	if(delta_poc_msb_present_flag[i])	
	delta_poc_msb_cycle_lt[i]	ue(v)
	used_by_curr_pic_lt_flag[i]	u(l)
	}	
	}	
	}	
	if(sample_adaptive_offset_enabled_flag) {	
	slice_sample_adaptive_offset_flag[0]	u(l)
	if(slice_sample_adaptive_offset_flag [0] {	
	slice_sample_adaptive_offset_flag[1]	u(l)
	slice_sample_adaptive_offset_flag[2]	u(l)
	}	
	}	

FIG. 16A

1614	if(adaptive_loop_filter_enabled_flag)	
	aps_id	ue(v)
1615	if(slice_type == P || slice_type== B) {	
1616	if(sps_temporal_mvp_enable_flag)	
	pic_temporal_mvp_enable_flag	u(l)
1617	num_ref_idx_active_override_flag	u(l)
1618	if(num_ref_idx_active_override_flag) {	
	num_ref_idx_10_active_minus1	ue(v)
	if(slice_type == B)	
	num_ref_idx_11_active_minus1	ue(v)
	}	
	}	
1619	if(lists_modification_present_flag)	
	ref_pic_list_modification( )	
1620	if(slice_type == B)	
	mvd_11_zero_flag	u(l)
1622	if(cabac_init_present_flag && slice_type != I)	
	cabac_init_flag	u(l)
1624	slice_qp_delta	se(v)
1626	if(deblocking_filter_control_present_flag) {	
	if (deblocking_filter_override_enabled_flag)	
	deblocking_filter_override_flag	u(l)
	if(deblocking_filter_override_flag) {	
	slice_header_disable_deblocking_filter_flag	u(l)
	if(!slice_header_disable_deblocking_filter_flag) {	
	beta_offset_div2	se(v)
	tc_offset_div2	se(v)
	}	
	}	
	}	
1628	if(pic_temporal_mvp_enable_flag) {	
1630	if(slice_type == B)	
1632	collocated_from_10_flag	u(l)
1634	if(slice_type !=I && ((collocated_from_10_flag && num_ref_idx_10_active_minus1 > 0) || (!collocated_from_10_flag && num_ref_idx_11_active_minus1 > 0))	
	collocated_ref_idx	ue(v)
	}	
1636	if((weighted_pred_flag && slice_type == P) || (weighted_bipred_idc == 1 && slice_type == B))	
	pred_weight_table( )	
1638	if(slice_type == P || slice_type == B)	
	five_minus_max_num_merge_cand	ue(v)
1640	if(adaptive_loop_filter_enabled_flag) {	
	slice_adaptive_loop_filter_flag	u(l)

FIG. 16B

1640	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_param( )	
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_cu_control_param( )	
	}	
1642	if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
	slice_loop_filter_across_slices_enabled_flag	u(l)
	}	
1644	if(tiles_or_entropy_coding_sync_idc = = 1 || tiles_or_entropy_coding_sync_idc = = 2) {	
	num_entry_point_offsets	ue(v)
	if(num_entry_point_offsets > 0) {	
	offset_len_minus1	ue(v)
	for(i=0; i< num_entry_point_offsets; i++)	
	entry_point_offset[i]	u(v)
	}	
	}	
	if(slice_header_extension_present_flag) {	
	slice_header_extension_length	ue(v)
	for(i=0; i < slide_header_extension_length; i++)	
	slice_header_extension_data_byte	u(8)
	}	
	byte_alignment()	
	}	

FIG. 16C

	aps_id	
1701	if(slice_type == P || slice_type== B) {	ue(v)
1702	if(sps_temporal_mvp_enable_flag)	
	pic_temporal_mvp_enable_flag	
1703	num_ref_idx_active_override_flag	u(l)
1704	if(num_ref_idx_active_override_flag) {	u(l)
	num_ref_idx_10_active_minus1	
	if(slice_type == B)	ue(v)
	num_ref_idx_11_active_minus1	
	}	ue(v)
1705	if(lists_modification_present_flag)	
	ref_pic_list_modification( )	
1706	if(slice_type == B)	
	mvd_11_zero_flag	u(l)
1708	if(cabac_init_present_flag && slice_type != I)	
	cabac_init_flag	u(l)
1710	if(pic_temporal_mvp_enable_flag) {	
	if(slice_type == B)	
	collocated_from_10_flag	u(l)
	if((collocated_from_10_flag && num_ref_idx_10_active_minus1 > 0) || (!collocated_from_10_flag && num_ref_idx_11_active_minus1 > 0))	
	collocated_ref_idx	ue(v)
	}	
1712	if((weighted_pred_flag && slice_type == P) || (weighted_bipred_idc == 1 && slice_type == B))	
	pred_weight_table( )	
1714	five_minus_max_num_merge_cand	ue(v)
	}	
1716	slice_qp_delta	se(v)
1718	if(deblocking_filter_control_present_flag) {	
	if (deblocking_filter_override_enabled_flag)	
	deblocking_filter_override_flag	u(l)
	if(deblocking_filter_override_flag) {	
	slice_header_disable_deblocking_filter_flag	u(l)
	if(!slice_header_disable_deblocking_filter_flag) {	
	beta_offset_div2	se(v)
	tc_offset_div2	se(v)
	}	
	}	

FIG. 17A

1720	if(adaptive_loop_filter_enabled_flag) {	
	slice_adaptive_loop_filter_flag	u(l)
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_param( )	
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_cu_control_param( )	
	}	
1722	if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
	slice_loop_filter_across_slices_enabled_flag	u(l)
	}	
1724	if(tiles_or_entropy_coding_sync_idc = = 1 || tiles_or_entropy_coding_sync_idc = = 2) {	
	num_entry_point_offsets	ue(v)
	if(num_entry_point_offsets > 0) {	
	offset_len_minus1	ue(v)
	for(i=0; i< num_entry_point_offsets; i++)	
	entry_point_offset[i]	u(v)
	}	
	}	
	if(slice_header_extension_present_flag) {	
	slice_header_extension_length	ue(v)
	for(i=0; i < slide_header_extension_length; i++)	
	slice_header_extension_data_byte	u(8)
	}	
	byte_alignment()	
	}	

FIG. 17B

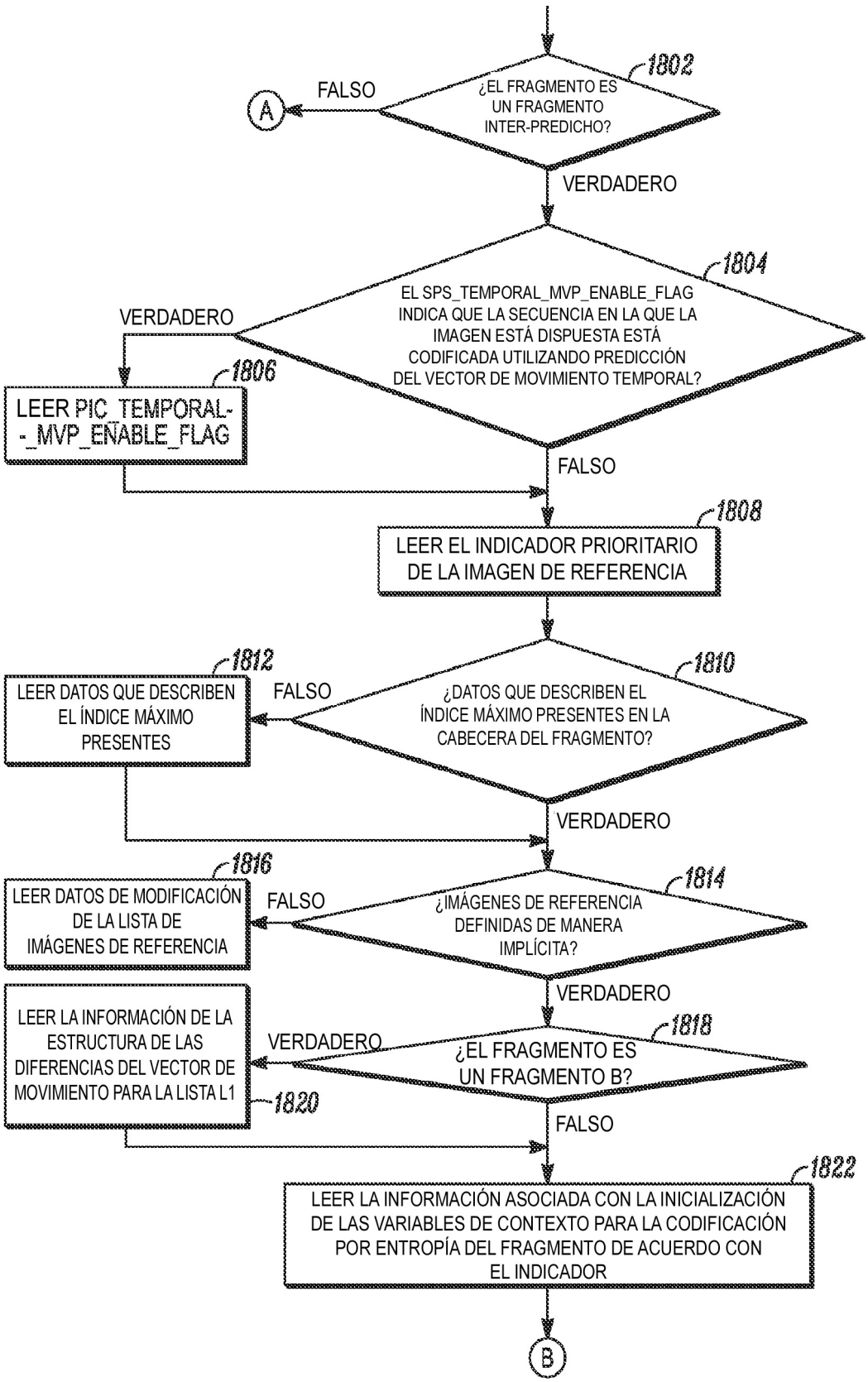

1802
¿EL FRAGMENTO ES UN FRAGMENTO INTER-PREDICHO?
FALSO
A
VERDADERO
1804
EL SPS_TEMPORAL_MVP_ENABLE_FLAG INDICA QUE LA SECUENCIA EN LA QUE LA IMAGEN ESTÁ DISPUESTA ESTÁ CODIFICADA UTILIZANDO PREDICCIÓN DEL VECTOR DE MOVIMIENTO TEMPORAL?
VERDADERO
1806
LEER PIC_TEMPORAL-_MVP_ENABLE_FLAG
FALSO
1808
LEER EL INDICADOR PRIORITARIO DE LA IMAGEN DE REFERENCIA
1810
¿DATOS QUE DESCRIBEN EL ÍNDICE MÁXIMO PRESENTES EN LA CABECERA DEL FRAGMENTO?
FALSO
1812
LEER DATOS QUE DESCRIBEN EL ÍNDICE MÁXIMO PRESENTES
VERDADERO
1814
¿IMÁGENES DE REFERENCIA DEFINIDAS DE MANERA IMPLÍCITA?
FALSO
1816
LEER DATOS DE MODIFICACIÓN DE LA LISTA DE IMÁGENES DE REFERENCIA
VERDADERO
1818
¿EL FRAGMENTO ES UN FRAGMENTO B?
VERDADERO
1820
LEER LA INFORMACIÓN DE LA ESTRUCTURA DE LAS DIFERENCIAS DEL VECTOR DE MOVIMIENTO PARA LA LISTA L1
FALSO
1822
LEER LA INFORMACIÓN ASOCIADA CON LA INICIALIZACIÓN DE LAS VARIABLES DE CONTEXTO PARA LA CODIFICACIÓN POR ENTROPÍA DEL FRAGMENTO DE ACUERDO CON EL INDICADOR
B

FIG. 18A

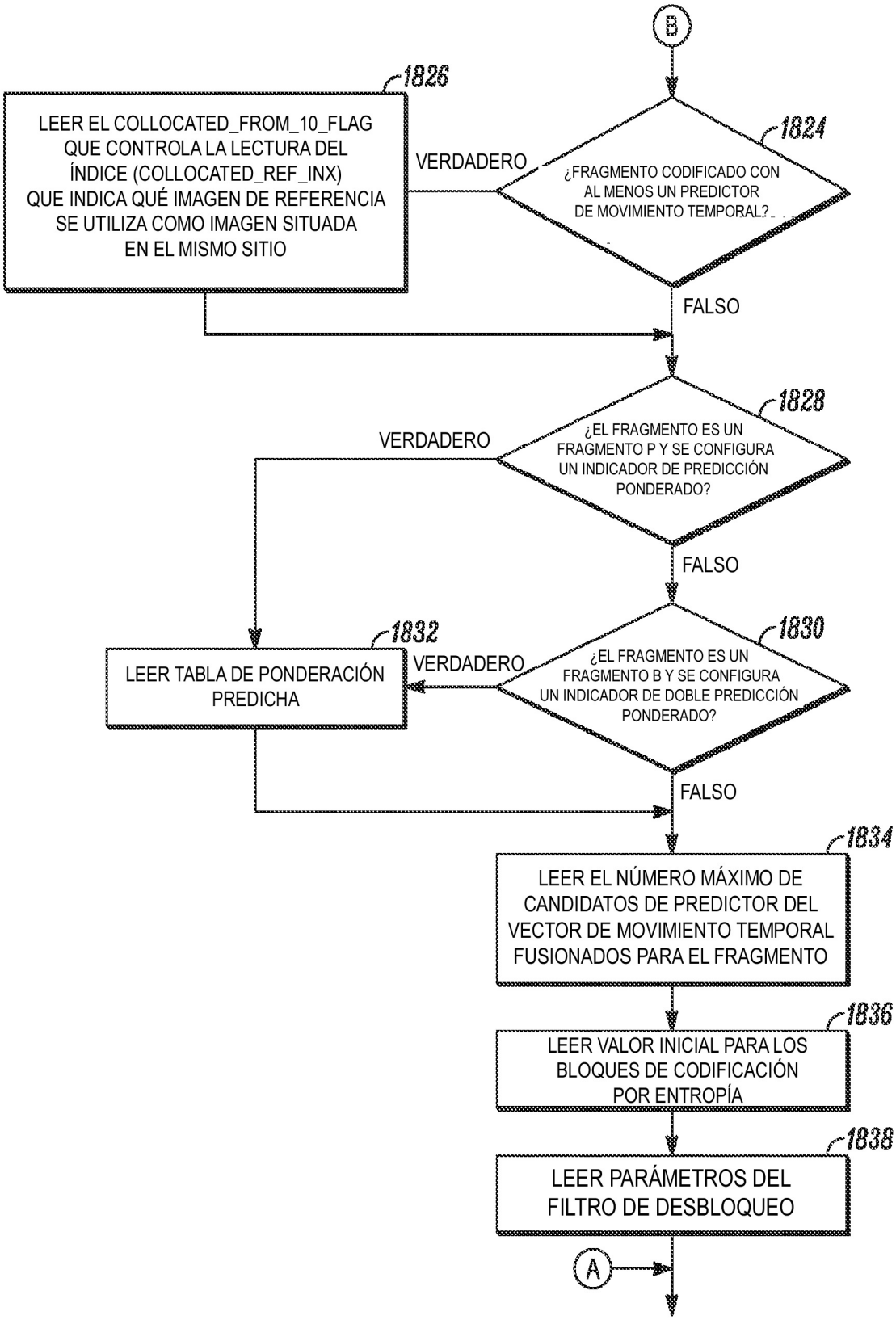

B
1826
LEER EL COLLOCATED_FROM_10_FLAG QUE CONTROLA LA LECTURA DEL ÍNDICE (COLLOCATED_REF_INX) QUE INDICA QUÉ IMAGEN DE REFERENCIA SE UTILIZA COMO IMAGEN SITUADA EN EL MISMO SITIO
VERDADERO
1824
¿FRAGMENTO CODIFICADO CON AL MENOS UN PREDICTOR DE MOVIMIENTO TEMPORAL?
FALSO
1828
¿EL FRAGMENTO ES UN FRAGMENTO P Y SE CONFIGURA UN INDICADOR DE PREDICCIÓN PONDERADO?
VERDADERO
FALSO
1830
¿EL FRAGMENTO ES UN FRAGMENTO B Y SE CONFIGURA UN INDICADOR DE DOBLE PREDICCIÓN PONDERADO?
1832
LEER TABLA DE PONDERACIÓN PREDICHA
VERDADERO
FALSO
1834
LEER EL NÚMERO MÁXIMO DE CANDIDATOS DE PREDICTOR DEL VECTOR DE MOVIMIENTO TEMPORAL FUSIONADOS PARA EL FRAGMENTO
1836
LEER VALOR INICIAL PARA LOS BLOQUES DE CODIFICACIÓN POR ENTROPÍA
1838
LEER PARÁMETROS DEL FILTRO DE DESBLOQUEO
A

FIG. 18B

	aps_id	
1716	slice_qp_delta	se(v)
	if(deblocking_filter_control_present_flag) {	
	if (deblocking_filter_override_enabled_flag)	
	deblocking_filter_override_flag	u(l)
	if(deblocking_filter_override_flag) {	
1718	slice_header_disable_deblocking_filter_flag	u(l)
	if(!slice_header_disable_deblocking_filter_flag) {	
	beta_offset_div2	se(v)
	tc_offset_div2	se(v)
	}	
	}	
	}	
1701	if(slice_type == P || slice_type== B) {	
	if(sps_temporal_mvp_enable_flag)	
1702	pic_temporal_mvp_enable_flag	
1703	num_ref_idx_active_override_flag	u(l)
	if(num_ref_idx_active_override_flag) {	u(l)
	num_ref_idx_10_active_minus1	
1704	if(slice_type == B)	ue(v)
	num_ref_idx_11_active_minus1	
	}	ue(v)
	if(lists_modification_present_flag)	
1705	ref_pic_list_modification( )	
	if(slice_type == B)	
1706	mvd_11_zero_flag	u(l)
	if(cabac_init_present_flag && slice_type != I)	
1708	cabac_init_flag	u(l)
	if(pic_temporal_mvp_enable_flag) {	
	if(slice_type == B)	
	collocated_from_10_flag	u(l)
1710	if((collocated_from_10_flag && num_ref_idx_10_active_minus1 > 0) ||	
	(!collocated_from_10_flag && num_ref_idx_11_active_minus1 > 0))	
	collocated_ref_idx	ue(v)
	}	
	if((weighted_pred_flag && slice_type == P) ||	
1712	(weighted_bipred_idc == 1 && slice_type == B))	
	pred_weight_table( )	
1714	five_minus_max_num_merge_cand	ue(v)
	}	

FIG. 19A

1720	if(adaptive_loop_filter_enabled_flag) {	
	slice_adaptive_loop_filter_flag	u(l)
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_param( )	
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_cu_control_param( )	
	}	
1722	if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
	slice_loop_filter_across_slices_enabled_flag	u(l)
	}	
1724	if(tiles_or_entropy_coding_sync_idc = = 1 || tiles_or_entropy_coding_sync_idc = = 2) {	
	num_entry_point_offsets	ue(v)
	if(num_entry_point_offsets > 0) {	
	offset_len_minus1	ue(v)
	for(i=0; i< num_entry_point_offsets; i++)	
	entry_point_offset[i]	u(v)
	}	

FIG. 19B

1701	if(slice_type == P || slice_type== B) {	ue(v)
1702	if(sps_temporal_mvp_enable_flag)	
	pic_temporal_mvp_enable_flag	
1703	num_ref_idx_active_override_flag	u(l)
1704	if(num_ref_idx_active_override_flag) {	u(l)
	num_ref_idx_10_active_minus1	
	if(slice_type == B)	ue(v)
	num_ref_idx_11_active_minus1	
	}	ue(v)
1705	if(lists_modification_present_flag)	
	ref_pic_list_modification( )	
1706	if(slice_type == B)	
	mvd_11_zero_flag	u(l)
1708	if(cabac_init_present_flag && slice_type != I)	
	cabac_init_flag	
	}	
1716	slice_qp_delta	se(v)
1718	if(deblocking_filter_control_present_flag) {	
	if (deblocking_filter_override_enabled_flag)	
	deblocking_filter_override_flag	u(l)
	if(deblocking_filter_override_flag) {	
	slice_header_disable_deblocking_filter_flag	u(l)
	if(!slice_header_disable_deblocking_filter_flag) {	
	beta_offset_div2	se(v)
	tc_offset_div2	se(v)
	}	
	}	
	}	

FIG. 20A

2002	if(slice_type ==P || slice_type ==B) {	
1710	if(pic_temporal_mvp_enable_flag) {	
	if(slice_type == B)	u(l)
	collocated_from_10_flag	
	if((collocated_from_10_flag && num_ref_idx_10_active_minus1 > 0) || (!collocated_from_10_flag && num_ref_idx_11_active_minus1 > 0))	
	collocated_ref_idx	ue(v)
	}	
1712	if((weighted_pred_flag && slice_type == P) ||	
	(weighted_bipred_idc == 1 && slice_type == B))	
	pred_weight_table( )	
1714	five_minus_max_num_merge_cand	ue(v)
	}	
1720	if(adaptive_loop_filter_enabled_flag) {	
	slice_adaptive_loop_filter_flag	u(l)
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_param( )	
	if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
	alf_cu_control_param( )	
	}	
1722	if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
	slice_loop_filter_across_slices_enabled_flag	u(l)
	}	
1724	if(tiles_or_entropy_coding_sync_idc = = 1 || tiles_or_entropy_coding_sync_idc = = 2) {	
	num_entry_point_offsets	ue(v)
	if(num_entry_point_offsets > 0) {	
	offset_len_minus1	ue(v)
	for(i=0; i< num_entry_point_offsets; i++)	
	entry_point_offset[i]	u(v)
	}	
	}	
	if(slice_header_extension_present_flag) {	
	slice_header_extension_length	ue(v)
	for(i=0; i < slide_header_extension_length; i++)	
	slice_header_extension_data_byte	u(8)
	}	
	byte_alignment()	
	}	

FIG. 20B

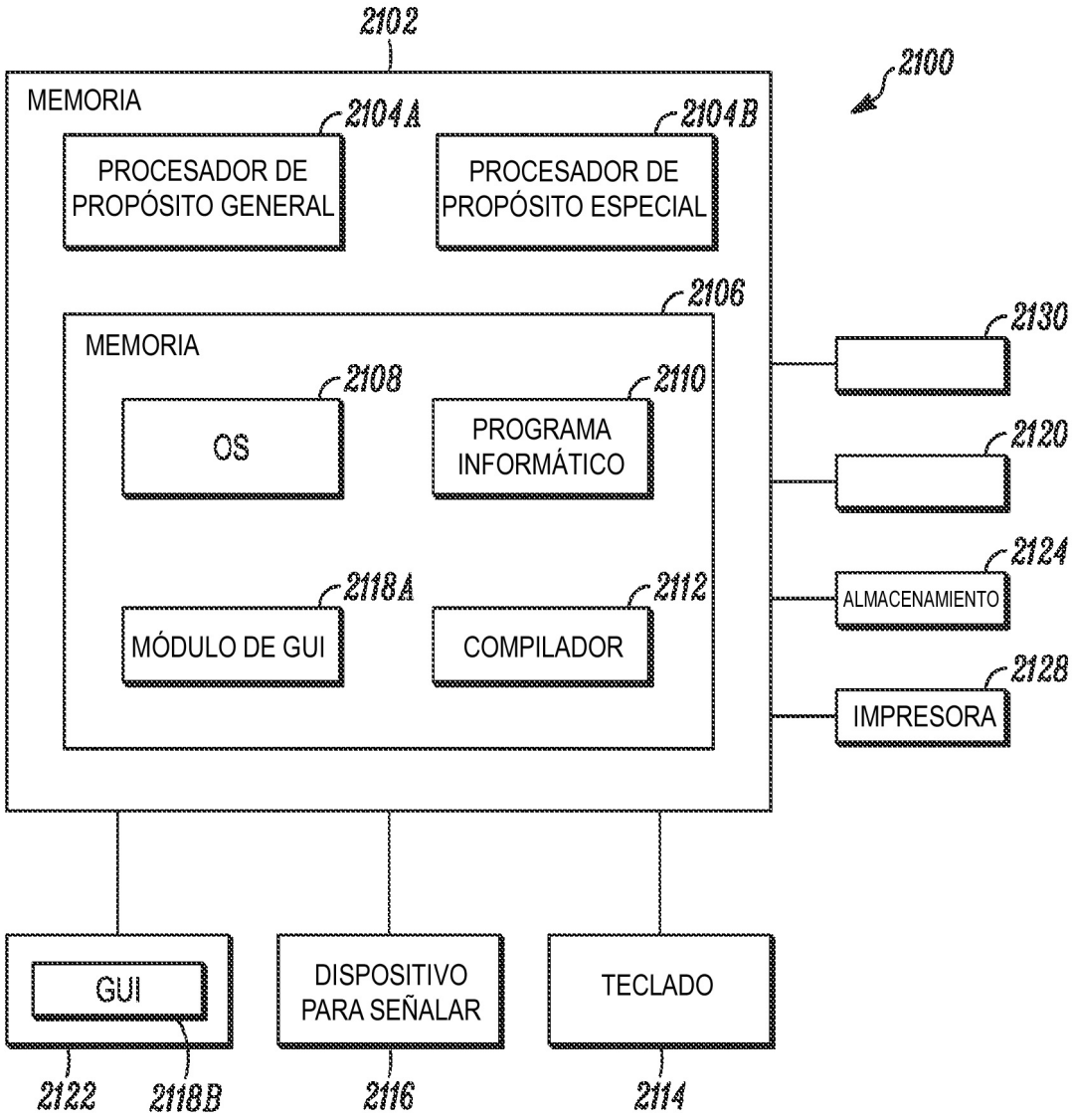

2102
2100
MEMORIA
2104A
2104B
PROCESADOR DE PROPÓSITO GENERAL
PROCESADOR DE PROPÓSITO ESPECIAL
2106
MEMORIA
2130
2108
2110
OS
PROGRAMA INFORMÁTICO
2120
2124
2118A
2112
ALMACENAMIENTO
MÓDULO DE GUI
COMPILADOR
2128
IMPRESORA
GUI
DISPOSITIVO PARA SEÑALAR
TECLADO
2122
2118B
2116
2114

FIG. 21