(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 693 707**

(51) Int. CI.:

H04N 19/597 (2014.01)
H04N 19/176 (2014.01)
H04N 19/70 (2014.01)
H04N 19/109 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **02.04.2013 PCT/US2013/034992**

(87) Fecha y número de publicación internacional: **17.10.2013 WO13154869**

(96) Fecha de presentación y número de la solicitud europea: **02.04.2013 E 13716683 (1)**

(97) Fecha y número de publicación de la concesión europea: **08.08.2018 EP 2837177**

(54) Título: **Modo de síntesis de visión para la codificación de vídeo tridimensional**

(30) Prioridad:

13.04.2012 US 201261624157 P
19.04.2012 US 201261635761 P
26.04.2012 US 201261639064 P
27.04.2012 US 201261639845 P
03.05.2012 US 201261642379 P
14.05.2012 US 201261646810 P
04.09.2012 US 201261696661 P
15.03.2013 US 201313839447

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
13.12.2018

(73) Titular/es:

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

(72) Inventor/es:

CHEN, YING;
ZHANG, LI;
YU, YANG;
VAN DER AUWERA, GEERT;
LI, XIANG y
KARCZEWICZ, MARTA

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Modo de síntesis de visión para la codificación de vídeo tridimensional

CAMPO TÉCNICO

[0001] Esta divulgación se refiere a la codificación de vídeo (es decir, a la codificación y/o decodificación de datos de vídeo).

ANTECEDENTES

[0002] Las capacidades del vídeo digital pueden incorporarse a una amplia gama de dispositivos, incluyendo televisores digitales, sistemas de radiodifusión digital directa, sistemas de radiodifusión inalámbrica, asistentes digitales personales (PDA), ordenadores portátiles o de escritorio, ordenadores tipo tablet, lectores de libros electrónicos, cámaras digitales, dispositivos de grabación digital, reproductores de medios digitales, dispositivos de videojuegos, consolas de videojuegos, teléfonos celulares o de radio por satélite, los denominados "teléfonos inteligentes", dispositivos de videoconferencia, dispositivos de transmisión de vídeo y similares. Los dispositivos de vídeo digitales implementan técnicas de compresión de vídeo, tales como las descritas en las normas definidas por MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Parte 10, Codificación Avanzada de Vídeo (AVC), la norma de Codificación de Vídeo de Alta Eficiencia (HEVC) actualmente en desarrollo y las ampliaciones de dichas normas. Los dispositivos de vídeo pueden transmitir, recibir, codificar, decodificar y/o almacenar información de vídeo digital más eficazmente implementando dichas técnicas de compresión de vídeo.

[0003] Las técnicas de compresión de vídeo ejecutan la predicción espacial (intraimagen) y/o la predicción temporal (interimagen) para reducir o eliminar la redundancia intrínseca en las secuencias de vídeo. Para la codificación de vídeo basada en bloques, un fragmento de vídeo (es decir, una trama de vídeo o una parte de una trama de vídeo) puede particionarse en bloques de vídeo. Los bloques de vídeo en un fragmento intracodificado (I) de una imagen se codifican mediante predicción espacial con respecto a muestras de referencia en bloques vecinos de la misma imagen. Los bloques de vídeo en un fragmento intercodificado (P o B) de una imagen pueden usar la predicción espacial con respecto a muestras de referencia en bloques vecinos de la misma imagen o la predicción temporal con respecto a muestras de referencia en otras imágenes de referencia. Las imágenes pueden denominarse tramas y las imágenes de referencia pueden denominarse tramas de referencia.

[0004] La predicción espacial o temporal da como resultado un bloque predictivo para un bloque que se vaya a codificar. Los datos residuales representan diferencias de píxeles entre el bloque original que se vaya a codificar y el bloque predictivo. Un bloque intercodificado se codifica de acuerdo con un vector de movimiento que apunta a un bloque de muestras de referencia que forman el bloque predictivo, y los datos residuales indican la diferencia entre el bloque codificado y el bloque predictivo. Un bloque intracodificado se codifica de acuerdo con un modo de intracodificación y los datos residuales. Para una mayor compresión, los datos residuales pueden transformarse desde el dominio del píxel en un dominio de transformada, dando como resultado coeficientes residuales, los cuales pueden cuantificarse posteriormente. Los coeficientes cuantificados, inicialmente dispuestos en una matriz bidimensional, pueden escanearse con el fin de producir un vector unidimensional de coeficientes, y puede aplicarse la codificación por entropía para lograr incluso más compresión.

[0005] Un flujo de bits de vistas múltiples puede generarse mediante la codificación de vistas por ejemplo, de múltiples cámaras a color. Para ampliar aún más la flexibilidad del vídeo de vistas múltiples, se han desarrollado normas de vídeo tridimensionales (3D). Un flujo de bits de vídeo en 3D puede contener no solo las vistas correspondientes a múltiples cámaras, concretamente vistas de textura, sino también vistas de profundidad asociadas con al menos una o más vistas de textura. Por ejemplo, cada vista puede consistir en una vista de textura y en una vista de profundidad.

[0006] Se hace referencia al siguiente artículo: "Mitsubishi Response to MPEG Call for Proposal on 3D Video Coding Technology" [“Respuesta de Mitsubishi a la Llamada de MPEG para la Propuesta sobre Tecnología de Codificación de Vídeo en 3D"], de DONG TIAN ET AL, 98. REUNIÓN DE MPEG; 28-11-2011 - 2-12-2011; GINEBRA; (GRUPO DE EXPERTOS DE IMÁGENES DE MOVIMIENTO O ISO/IEC JTC1/SC29/WG11),, n. º m22663, 24 de noviembre de 2011 (2011-11-24), XP030051226. El documento US 2008/198924 A1 describe un procedimiento para calcular la disparidad, el procedimiento de sintetización de la vista de interpolación, el procedimiento de codificación y decodificación de vídeo de vistas múltiples que usan el mismo, y el codificador y el decodificador que usan el mismo. El documento EP 1978750 A2 describe un procedimiento y un sistema para procesar vídeos de vistas múltiples para la síntesis de vistas usando modos de omisión y directos.

SUMARIO

[0007] La invención se define en las reivindicaciones a las que se dirige ahora la referencia. En general, esta divulgación describe la señalización de un modo de predicción de síntesis de visión (VSP) de una unidad de vídeo. Más específicamente, un codificador de vídeo señala, en un flujo de bits, un elemento de sintaxis que indica si una unidad de vídeo actual se predice a partir de una imagen VSP. La unidad de vídeo actual puede ser un macrobloque

o una partición de macrobloque. Además, el codificador de vídeo puede determinar, basándose al menos en parte en si la unidad de vídeo actual se predice a partir de la imagen VSP, si señalar, en el flujo de bits, información de movimiento para la unidad de vídeo actual.

[0008] Un decodificador de vídeo puede decodificar, a partir de un flujo de bits, el elemento de sintaxis que indica si la unidad de vídeo actual se predice a partir de la imagen VSP. Cuando la unidad de vídeo actual se predice a partir de la imagen VSP, el decodificador de vídeo puede reconstruir, basándose al menos en parte en la imagen VSP, bloques de muestra de la unidad de vídeo actual. Cuando la unidad de vídeo actual no se predice a partir de la imagen VSP, el decodificador de vídeo puede decodificar, a partir del flujo de bits, la información de movimiento para la unidad de vídeo actual. El decodificador de vídeo puede usar la información de movimiento para reconstruir los bloques de muestra de la unidad de vídeo actual.

[0009] Los detalles de uno o más ejemplos de la divulgación se exponen en los dibujos adjuntos y en la descripción siguiente. Otras características, objetivos y ventajas resultarán evidentes a partir de la descripción, de los dibujos y de las reivindicaciones.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0010]

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

La FIG. 2 es un diagrama de bloques que ilustra un codificador de vídeo de ejemplo que puede implementar las técnicas descritas en esta divulgación.

La FIG. 3 es un diagrama de bloques que ilustra un decodificador de vídeo de ejemplo que puede implementar las técnicas descritas en esta divulgación.

La FIG. 4A es un diagrama de flujo que ilustra una operación de ejemplo de un codificador de vídeo, de acuerdo con una o más técnicas de esta divulgación.

La FIG. 4B es un diagrama de flujo que ilustra una operación de ejemplo de un decodificador de vídeo, de acuerdo con una o más técnicas de esta divulgación.

La FIG. 5A es un diagrama de flujo que ilustra otra operación de ejemplo del codificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 5B es un diagrama de flujo que ilustra otra operación de ejemplo del decodificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 6A es un diagrama de flujo que ilustra otra operación de ejemplo del codificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 6B es un diagrama de flujo que ilustra otra operación de ejemplo del decodificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 7A es un diagrama de flujo que ilustra otra operación de ejemplo del codificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 7B es un diagrama de flujo que ilustra otra operación de ejemplo del decodificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 8A es un diagrama de flujo que ilustra otra operación de ejemplo del codificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 8B es un diagrama de flujo que ilustra otra operación de ejemplo del decodificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 9A es un diagrama de flujo que ilustra otra operación de ejemplo del codificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 9B es un diagrama de flujo que ilustra otra operación de ejemplo del decodificador de vídeo, de acuerdo con las técnicas de esta divulgación.

La FIG. 10 es un diagrama conceptual que ilustra un orden de decodificación de vídeo tridimensional (3DV) de ejemplo.

La FIG. 11 es un diagrama conceptual que ilustra una estructura de predicción temporal e inter-vistas de ejemplo.

DESCRIPCIÓN DETALLADA

[0011] En la codificación de vídeo tridimensional (3DV), las imágenes de la misma escena se capturan desde diferentes puntos de vista. Mostrar imágenes de la misma escena desde diferentes puntos de vista puede proporcionar al visor un efecto estereoscópico tridimensional. Debido a que las imágenes de la misma escena capturadas al mismo tiempo desde diferentes puntos de vista pueden ser muy similares, un codificador de vídeo puede usar una predicción inter-vistas para reducir la cantidad de datos enviados al predecir bloques de imágenes basadas en bloques en otras imágenes desde diferentes puntos de vista. El término "unidad de acceso" se usa para referirse al conjunto de imágenes que corresponden a la misma instancia de tiempo. Un "componente de vista" puede ser una representación codificada de una vista en una única unidad de acceso.

[0012] Para reducir más la cantidad de datos enviados, el codificador de vídeo puede generar una predicción de síntesis de visión (VSP) basándose en componentes de vista codificados previamente en la misma unidad de acceso como una imagen que se esté codificando actualmente. El codificador de vídeo puede incluir la imagen VSP en una lista de imágenes de referencia. Cuando el codificador de vídeo codifica una unidad de vídeo actual (por ejemplo, un macrobloque (MB), una partición de MB, una partición de subMB, una unidad de predicción (PU), etc.), el codificador de vídeo puede usar la imagen VSP como una imagen de referencia al generar un bloque predictivo para la unidad de vídeo actual. Además, el codificador de vídeo puede señalar un índice de referencia y un vector de movimiento. El índice de referencia puede indicar la posición de la imagen VSP dentro de la lista de imágenes de referencia. El vector de movimiento indica un desplazamiento espacial entre un bloque de referencia dentro de la imagen VSP y bloques de muestra de la unidad de vídeo actual.

[0013] El codificador de vídeo puede señalar el vector de movimiento usando una diferencia de vector de movimiento (MVD). La MVD puede indicar una diferencia entre un predictor de vector de movimiento y un vector de movimiento de la unidad de vídeo actual. El predictor de vector de movimiento puede ser un vector de movimiento de un bloque vecino.

[0014] Un decodificador de vídeo puede generar la misma imagen VSP como el codificador de vídeo y puede generar la misma lista de imágenes de referencia como el codificador de vídeo. Además, el decodificador de vídeo puede determinar, basándose en el índice de referencia, que se debe generar un bloque predictivo para una unidad de vídeo actual basándose en la imagen VSP. Además, el decodificador de vídeo puede determinar, basándose al menos en parte en una MVD señalizada, un vector de movimiento para la unidad de vídeo actual. El decodificador de vídeo puede determinar entonces, basándose al menos en parte en el vector de movimiento, un bloque de referencia dentro de la imagen VSP. A continuación, el decodificador de vídeo puede determinar, basándose al menos en parte en el bloque de referencia, el bloque predictivo para la unidad de vídeo actual. El decodificador de vídeo puede reconstruir, basándose al menos en parte en el bloque predictivo para la unidad de vídeo actual, bloques de muestra de la unidad de vídeo actual.

[0015] Como se mencionó anteriormente, cuando el codificador de vídeo usa una imagen VSP para generar un bloque de predicción para una unidad de vídeo actual, el codificador de vídeo señala una MVD desde la que el decodificador de vídeo deriva un vector de movimiento de la unidad de vídeo actual. Cuando el codificador de vídeo usa una imagen VSP para generar un bloque predictivo para la unidad de vídeo actual, el vector de movimiento casi siempre está muy cerca de cero. Es decir, el bloque de referencia en la imagen VSP casi siempre está ubicado junto con los bloques de muestra de la unidad de vídeo actual.

[0016] Debido a que el bloque de referencia en la imagen VSP está casi siempre situado con los bloques de muestras de la unidad de vídeo actual, el decodificador de vídeo puede ser capaz de determinar, sin decodificar una MVD para la unidad de vídeo actual del flujo de bits, que un vector de movimiento de la unidad de vídeo actual es igual a 0 si la unidad de vídeo actual se codifica basándose en la imagen VSP. Por lo tanto, puede ser un desperdicio de bits señalar una MVD para la unidad de vídeo actual cuando la unidad de vídeo actual se codifique basándose en la imagen VSP. Además, como no es necesario señalar la MVD para la unidad de vídeo actual cuando la unidad de vídeo actual se codifique basándose en la imagen VSP, la señalización de un elemento de sintaxis que indique que la unidad de vídeo actual se codifica basándose en la imagen VSP puede no hacer necesario incluir la imagen VSP en una lista de imágenes de referencia.

[0017] De acuerdo con las técnicas de esta divulgación, el codificador de vídeo puede señalar, en un flujo de bits que incluya una representación codificada de vistas múltiples de textura y vistas múltiples de profundidad, un elemento de sintaxis que indique si una unidad de vídeo actual se predice a partir de una imagen VSP de un componente de vista de textura actual. En algunos ejemplos, la unidad de vídeo actual puede ser un MB, una partición de MB o una partición de subMB. En otros ejemplos, la unidad de vídeo actual puede ser una unidad de predicción (PU). En algunos ejemplos, cuando el elemento de sintaxis indica que la unidad de vídeo actual se predice a partir de la imagen VSP, el codificador

de vídeo no señala información de movimiento para la unidad de vídeo actual. En otras palabras, el codificador de vídeo omite, del flujo de bits, la información de movimiento para la unidad de vídeo actual. Por ejemplo, el codificador de vídeo no señala índices de referencia o MVD para la unidad de vídeo actual cuando la unidad de vídeo actual se predice a partir de la imagen VSP. Por el contrario, cuando la unidad de vídeo actual no se predice a partir de la imagen VSP, el codificador de vídeo puede señalar, en el flujo de bits, la información de movimiento para la unidad de vídeo actual. Por tanto, el codificador de vídeo puede determinar, basándose al menos en parte en si la unidad de vídeo actual se predice a partir de la imagen VSP, si señalar, en el flujo de bits, información de movimiento para la unidad de vídeo actual. Al no señalar la información de movimiento de la unidad de vídeo actual cuando se prediga la unidad de vídeo actual a partir de la imagen VSP, el codificador de vídeo puede reducir el número de bits en el flujo de bits.

[0018] De forma similar, de acuerdo con las técnicas de la presente divulgación, un decodificador de vídeo puede generar, basándose al menos en parte en un componente de vista de textura codificado previamente de una unidad de acceso actual y en un componente de vista de profundidad de la unidad de acceso actual, una imagen VSP. Además, el decodificador de vídeo puede decodificar, a partir de un flujo de bits que incluya una representación codificada de vistas múltiples de textura y vistas múltiples de profundidad, un elemento de sintaxis que indique si se predice una unidad de vídeo actual a partir de la imagen VSP. Cuando la unidad de vídeo actual no se prediga a partir de la imagen VSP, el decodificador de vídeo puede decodificar, desde el flujo de bits, información de movimiento para la unidad de vídeo actual y puede usar la información de movimiento para la unidad de vídeo actual para reconstruir bloques de muestra de la unidad de vídeo actual. Cuando la unidad de vídeo actual se predice a partir de la imagen VSP, el decodificador de vídeo puede usar la imagen VSP para reconstruir los bloques de muestra de la unidad de vídeo actual.

[0019] La FIG. 1 es un diagrama de bloques que ilustra un sistema de codificación de vídeo 10 de ejemplo que puede utilizar las técnicas de esta divulgación. Como se usa en el presente documento, el término "codificador de vídeo" se refiere genéricamente tanto a codificadores de vídeo como a decodificadores de vídeo. En esta divulgación, los términos "codificación de vídeo" o "codificación" pueden referirse genéricamente a la codificación de vídeo o a la decodificación de vídeo.

[0020] Como se muestra en la figura. 1, el sistema de codificación de vídeo 10 incluye un dispositivo de origen 12 y un dispositivo de destino 14. El dispositivo de origen 12 genera datos de vídeo codificados. En consecuencia, el dispositivo de origen 12 puede denominarse dispositivo de codificación de vídeo o aparato de codificación de vídeo. El dispositivo de destino 14 puede decodificar datos de vídeo codificados, generados por el dispositivo de origen 12. En consecuencia, el dispositivo de destino 14 puede denominarse dispositivo de decodificación de vídeo o aparato de decodificación de vídeo. El dispositivo de origen 12 y el dispositivo de destino 14 pueden ser ejemplos de dispositivos de codificación de vídeo o aparatos de codificación de vídeo.

[0021] El dispositivo de origen 12 y el dispositivo de destino 14 pueden comprender una amplia variedad de dispositivos, incluyendo ordenadores de sobremesa, dispositivos informáticos móviles, notebooks (es decir, portátiles), ordenadores tipo tablet, decodificadores, equipos telefónicos portátiles tales como los denominados teléfonos "inteligentes", televisores, cámaras, dispositivos de visualización, reproductores de medios digitales, consolas de videojuegos, ordenadores de coche o similares.

[0022] El dispositivo de destino 14 puede recibir datos de vídeo codificados desde el dispositivo de origen 12 a través de un canal 16. El canal 16 puede comprender uno o más medios o dispositivos capaces de desplazar los datos de vídeo codificados desde el dispositivo de origen 12 al dispositivo de destino 14. En un ejemplo, el canal 16 puede comprender uno o más medios de comunicación que permitan al dispositivo de origen 12 transmitir datos de vídeo codificados directamente al dispositivo de destino 14 en tiempo real. En este ejemplo, el dispositivo de origen 12 puede modular los datos de vídeo codificados de acuerdo con una norma de comunicación, tal como un protocolo de comunicación inalámbrica, y puede transmitir los datos de vídeo modulados al dispositivo de destino 14. Los uno o más medios de comunicación pueden incluir medios de comunicación inalámbricos y/o alámbricos, tales como un espectro de radiofrecuencia (RF) o una o más líneas de transmisión física. Los uno o más medios de comunicación pueden formar parte de una red basada en paquetes, tal como una red de área local, una red de área extensa o una red global (por ejemplo, Internet). Los uno o más medios de comunicación pueden incluir routers, conmutadores, estaciones base u otros equipos que faciliten la comunicación desde el dispositivo de origen 12 al dispositivo de destino 14.

[0023] En otro ejemplo, el canal 16 puede incluir un medio de almacenamiento que almacene los datos de vídeo codificados generados por el dispositivo de origen 12. En este ejemplo, el dispositivo de destino 14 puede acceder al medio de almacenamiento a través del acceso al disco o del acceso a la tarjeta. El medio de almacenamiento puede incluir varios medios de almacenamiento de datos de acceso local tales como discos Blu-ray, DVD, CD-ROM, memoria flash u otros medios adecuados de almacenamiento digital para almacenar datos de vídeo codificados.

[0024] En un ejemplo adicional, el canal 16 puede incluir un servidor de archivos u otro dispositivo de almacenamiento intermedio que almacene los datos de vídeo codificados generados por el dispositivo de origen 12. En este ejemplo, el dispositivo de destino 14 puede acceder a datos de vídeo codificados almacenados en el servidor de ficheros o en otro dispositivo de almacenamiento intermedio mediante transmisión continua o descarga. El servidor de ficheros

puede ser un tipo de servidor capaz de almacenar datos de vídeo codificados y transmitir los datos de vídeo codificados al dispositivo de destino 14. Entre los ejemplos de servidores de ficheros se incluyen servidores de red (por ejemplo, para una página web), servidores del protocolo de transferencia de ficheros (FTP), dispositivos de almacenamiento conectados a red (NAS) y unidades de disco local.

[0025] El dispositivo de destino 14 puede acceder a los datos de vídeo codificados a través de una conexión de datos estándar, tal como una conexión a Internet. Entre los ejemplos de tipos de conexiones de datos pueden incluirse canales inalámbricos (por ejemplo, conexiones WiFi), conexiones cableadas (por ejemplo, DSL, módem de cable, etc.) o combinaciones de ambos que sean adecuadas para acceder a datos de vídeo codificados almacenados en un servidor de ficheros. La transmisión de datos de vídeo codificados desde el servidor de ficheros puede ser una transmisión continua, una transmisión de descarga o una combinación de ambas.

[0026] Las técnicas de esta divulgación no están limitadas a aplicaciones o a configuraciones inalámbricas. Las técnicas pueden aplicarse a la codificación de vídeo en soporte de una diversidad de aplicaciones multimedia, tales como radiodifusiones de televisión por el aire, radiodifusiones de televisión por cable, transmisiones de televisión por satélite, transmisiones de vídeo en continuo, por ejemplo, mediante Internet, codificación de datos de vídeo para su almacenamiento en un medio de almacenamiento de datos, decodificación de datos de vídeo almacenados en un medio de almacenamiento de datos, u otras aplicaciones. En algunos ejemplos, el sistema de codificación de vídeo 10 puede configurarse para soportar la transmisión de vídeo unidireccional o bidireccional para soportar aplicaciones tales como la transmisión de vídeo, la reproducción de vídeo, la radiodifusión de vídeo y/o la videotelefonía.

[0027] En el ejemplo de la FIG. 1, el dispositivo de origen 12 incluye una fuente de vídeo 18, un codificador de vídeo 20 y una interfaz de salida 22. En algunos ejemplos, la interfaz de salida 22 puede incluir un modulador/demodulador (módem) y/o un transmisor. La fuente de vídeo 18 puede incluir un dispositivo de captura de vídeo, por ejemplo, una videocámara, un fichero de vídeo que contenga datos de vídeo previamente capturados, una interfaz de alimentación de vídeo para recibir datos de vídeo desde un proveedor de contenido de vídeo y/o un sistema de gráficos por ordenador para generar datos de vídeo, o una combinación de dichas fuentes de datos de vídeo.

[0028] El codificador de vídeo 20 puede codificar datos de vídeo de la fuente de vídeo 18. En algunos ejemplos, el dispositivo de origen 12 transmite directamente los datos de vídeo codificados al dispositivo de destino 14 mediante la interfaz de salida 22. En otros ejemplos, los datos de vídeo codificados también pueden almacenarse en un medio de almacenamiento o en un servidor de ficheros para un acceso posterior mediante el dispositivo de destino 14 para su decodificación y/o reproducción.

[0029] En el ejemplo de la FIG. 1, el dispositivo de destino 14 incluye una interfaz de entrada 28, un decodificador de vídeo 30 y un dispositivo de visualización 32. En algunos ejemplos, la interfaz de entrada 28 incluye un receptor y/o un módem. La interfaz de entrada 28 puede recibir los datos de vídeo codificados por el canal 16. El dispositivo de visualización 32 puede estar integrado con, o ser externo a, el dispositivo de destino 14. En general, el dispositivo de visualización 32 muestra los datos de vídeo decodificados. El dispositivo de visualización 32 puede comprender varios dispositivos de visualización, tales como una pantalla de cristal líquido (LCD), una pantalla de plasma, una pantalla de diodos orgánicos emisores de luz (OLED) u otro tipo de dispositivo de visualización.

[0030] En algunos ejemplos, el codificador de vídeo 20 y el decodificador de vídeo 30 funcionan de acuerdo con una norma de compresión de vídeo, tal como ISO/IEC MPEG-4 Visual e ITU-T H.264 (también conocida como ISO/IEC MPEG-4 AVC), incluidas sus extensiones de codificación de vídeo escalable (SVC) y de codificación de vídeo de vistas múltiples (MVC). Un borrador de MVC se describe en "Advanced vídeo coding for generic audiovisual services, Recomendación UIT-T H.264, marzo de 2010, que a partir del 13 de marzo de 2013 está disponible para su descarga desde http://www.itu.int/rec. /T-REC-H.264-201003-S/en. Otro borrador reciente de la extensión de MVC de H.264/AVC está disponible desde el 13 de marzo de 2013 para su descarga en http://wftp3.itu.int/av-arch/jvt-site/2009_01_Geneva/JVTAD007.zip.

[0031] Además, hay una extensión de la norma de MVC, concretamente "MVC-based 3DV" ["3DV basada en MVC"] (es decir, 3DV compatible con MVC), descrito en "WD of MVC extension for inclusion of depth maps", documento w12351 de MPEG. En algunos casos, cualquier flujo de bits que se ajuste a la 3DV basada en MVC siempre contiene un subflujo de bits que es compatible con un perfil de MVC, por ejemplo, un alto perfil estéreo.

[0032] Además, actualmente se están emprendiendo unas iniciativas para generar una ampliación de codificación de vídeo tridimensional (3DV) para H.264/AVC, en concreto, 3DV basada en AVC. Un borrador de trabajo (WD) de 3DV basada en AVC se denomina "Borrador de Trabajo 1 de 3DV-AVC". Otro borrador de 3DV basada en AVC se describe en Mannuksela et al., "Texto Borrador 4 de 3D-AVC", Equipo conjunto de colaboración en el desarrollo de la extensión de codificación de vídeo en 3D de ITU-T SG 16 WP 3 e ISO/IEC JTC 1/SC 29/WG 11, segunda reunión, Shanghai China, octubre de 2012, que, desde el 13 de marzo de 2013, está disponible en http: //phenix.it-sudparis.eu/jct2/doc_end_user/documents/2_Shanghai/wgll/JCT3V-B1002-vl.zip. Una descripción del software de referencia para 3DV basada en AVC está disponible como Miska M. Hannuksela, "Test Model for AVC based 3D vídeo coding" ["Modelo de Prueba para la codificación de vídeo en 3D basada en AVC"], ISO/IEC JTC1/SC29/WG11

MPEG2011/N12558, San José, EE.UU., febrero de 2012. El software de referencia está disponible desde el 13 de marzo de 2013 en http://mpeg3dv.research.nokia.com/svn/mpeg3dv/trunk/.

[0033] En otros ejemplos, el codificador de vídeo 20 y el decodificador de vídeo 30 pueden funcionar de acuerdo con la ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 o ISO/IEC MPEG-2 Visual e ITU-T H.264, ISO/IEC Visual. El codificador de vídeo 20 y el decodificador de vídeo 30 pueden funcionar de acuerdo con una norma de compresión de vídeo, tal como la norma de Codificación de Vídeo de Alta Eficacia (HEVC), actualmente en fase de desarrollo por parte del Equipo de Colaboración Conjunta sobre Codificación de vídeo (JCT-VC) del Grupo de Expertos de Codificación de Vídeo (VCEG) de la UIT-T y del Grupo de Expertos en Imágenes en Movimiento (MPEG) de ISO/IEC.

[0034] Un borrador de la próxima norma de HEVC, denominado "Borrador de Trabajo 4 de la HEVC", se describe en Bross et al., "WD4: Working Draft 4 of High Efficiency Video Coding" ["Borrador de Trabajo 4 de la HEVC"], Equipo conjunto de colaboración en codificación de vídeo (JCT-VC) de ITUT SG16 WP3 e ISO/IEC JTC1/SC29/WG11, 6ª reunión, Torino, IT, julio de 2011, que, a partir de marzo 13, 2013, está disponible en http://phenix.int-evry.fr/jct/doc_end_user/documents/6_Torino/wg11/JCTVC-F803-v8.zip. Otro borrador de la próxima norma de HEVC, conocido como "Borrador de Trabajo 6 de la HEVC" se describe en Bross et al., "High Efficiency Video Coding (HEVC) text specification draft 6" ["Borrador de Trabajo 6 de la HEVC" se describe en Bross et al., Borrador 6 de la memoria descriptiva del texto de Codificación de Vídeo de Alta Eficiencia (HEVC)"], Equipo colaborativo conjunto en codificación de vídeo (JCT-VC) de UIT-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, 8ª reunión, San José, CA, febrero de 2012, que, a partir del 13 de marzo de 2013 está disponible en http://phenix.int-evry.fr /jct/doc_end_user/documents/8_San%20Jose/wg11/JCTVC-H1003-v22.zip. Otro borrador de la siguiente norma de HEVC, denominado “Borrador de Trabajo 9 de la HEVC”, se describe en el documento de Bross et al. "High Efficiency Video Coding (HEVC) text specification draft 9" ["Borrador de Trabajo 9 de la HEVC" se describe en Bross et al., Borrador 6 de la memoria descriptiva del texto de Codificación de Vídeo de Alta Eficiencia (HEVC)"], Equipo de Colaboración Conjunta en Codificación de Vídeo (JCT-VC) de ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, 11.ª reunión: Shanghai, China, octubre de 2012, que, desde el 13 de marzo de 2013, se puede descargar desde http://phenix.int-evry.fr/jct/doc_end_user/documents/11_Shanghai/wg11/JCTVC-K1003-v8.zip.

[0035] Además, se están haciendo esfuerzos para producir una extensión de 3DV para la HEVC. La extensión de 3DV de la HEVC puede denominarse 3DV basada en HEVC o HEVC-3DV. El códec 3DV basado en HEVC en el MPEG se basa en las soluciones propuestas en Schwarz et al., "Description of 3D Video Technology Proposal by Fraunhofer HHI (HEVC compatible; configuración A)" ["Descripción de la Propuesta de Tecnología de Vídeo en 3D de Fraunhofer HHI (HEVC compatible; configuración A)"], ISO/IEC JTC1/SC29/WG11 MPEG2011/m22570, Ginebra, Suiza, noviembre de 2011 (en adelante, "documento m22570") y Wegner et al., "Poznan University of Technology tools for 3DV coding integrated into 3D-HTM" ["Universidad de Poznan de herramientas de Tecnología para la codificación de 3DV integrada en 3D-HTM"], ISO/IEC JTC1/SC29/WG11 MPEG2011/m23783, San José, EE.UU., Febrero de 2012 (en adelante, "documento m23783"). Una descripción de software de referencia está disponible en Schwarz y col., “Test Model under Consideration for HEVC based 3D vídeo coding” [“Modelo de Prueba Considerado para la codificación de vídeo en 3D basada en HEVC”], ISO/IEC JTC1/SC29/WG11 MPEG2011/N12559, San José, Estados Unidos, febrero de 2012. El software de referencia está disponible, desde el 13 de marzo de 2013, en https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/trunk. Aunque las técnicas de esta divulgación se describen con respecto a H.264/AVC y HEVC, dichas técnicas pueden aplicarse a otras normas de codificación y no están limitadas a ninguna norma o técnica de codificación particular.

[0036] La FIG. 1 es meramente un ejemplo y las técnicas de esta divulgación pueden aplicarse a ajustes de codificación de vídeo (por ejemplo, codificación de vídeo o decodificación de vídeo) que no incluyen necesariamente ninguna comunicación de datos entre los dispositivos de codificación y de decodificación. En otros ejemplos, los datos se recuperan de una memoria local, se transmiten a través de una red o similar. Un dispositivo de codificación de vídeo puede codificar y almacenar datos en la memoria, y/o un dispositivo de decodificación de vídeo puede recuperar y decodificar datos desde la memoria. En muchos ejemplos, la codificación y la decodificación se realiza mediante dispositivos que no se comunican entre sí, sino que simplemente codifican datos en la memoria y/o recuperan y decodifican datos desde la memoria.

[0037] El codificador de vídeo 20 y el decodificador de vídeo 30 pueden implementarse como cualquiera entre una variedad de circuitos adecuados, tales como uno o más microprocesadores, procesadores de señales digitales (DSP), circuitos integrados específicos de la aplicación (ASIC), matrices de puertas programables por campo (FPGA), lógica discreta, hardware o cualquier combinación de los mismos. Si las técnicas se implementan parcialmente en software, un dispositivo puede almacenar instrucciones para el software en un medio adecuado de almacenamiento no transitorio legible por ordenador y puede ejecutar las instrucciones en hardware usando uno o más procesadores para realizar las técnicas de esta divulgación. Cualquiera de los anteriores (incluyendo hardware, software, una combinación de hardware y software, etc.) puede considerarse como uno o más procesadores. Cada uno del codificador de vídeo 20 y del decodificador de vídeo 30 pueden estar incluidos en uno o más codificadores o decodificadores, cualquiera de los cuales puede estar integrado como parte de un codificador/decodificador (CÓDEC) combinado en un dispositivo respectivo.

[0038] Esta divulgación puede referirse en general al codificador de vídeo 20 que "señala" cierta información a otro dispositivo, tal como el decodificador de vídeo 30. El término "señalar" puede referirse en general a la comunicación de elementos de sintaxis y/o a otros datos usados para decodificar los datos de vídeo comprimidos. Dicha comunicación puede producirse en tiempo real o casi real. De forma alternativa, dicha comunicación puede producirse durante un tramo de tiempo, tal como podría ocurrir cuando se almacenen elementos de sintaxis en un medio de almacenamiento legible por ordenador en un flujo de bits codificado en el momento de la codificación, que a continuación pueda recuperarse por un dispositivo de decodificación en cualquier momento tras haberse almacenado en este medio.

[0039] Una secuencia de vídeo incluye típicamente una serie de imágenes. Las imágenes también pueden denominarse "tramas". Una imagen puede incluir tres matrices de muestra, indicadas con S_L, S_{Cb} y S_{Cr}. S_L es una matriz bidimensional (es decir, un bloque) de muestras de luminancia. S_{Cb} es una matriz bidimensional de muestras de crominancia Cb. S_{Cr} es una matriz bidimensional de muestras de crominancia Cr. Las muestras de crominancia también se pueden denominar en el presente documento muestras de "crominancia". En otros casos, una imagen puede ser monocromática y puede solo incluir una matriz de muestras de luminancia.

[0040] Para generar una representación codificada de una imagen, el codificador de vídeo 20 puede dividir las matrices de muestra de la imagen en bloques de igual tamaño. Por ejemplo, en H.264/AVC, el codificador de vídeo 20 puede dividir la imagen en macrobloques (MB). Un MB es un bloque 16x16 de muestras de luminancia y dos bloques correspondientes de muestras de crominancia de una imagen que tiene tres matrices de muestra, o un bloque 16x16 de muestras de una imagen monocromática o una imagen que se codifica mediante tres planos de color separados. En H.264/AVC, un fragmento puede incluir un número entero de MB o de pares de MB ordenados consecutivamente en el escaneo de barrido dentro de un grupo de fragmentos en particular.

[0041] El codificador de vídeo 20 puede particionar un MB en un conjunto de una o más particiones de MB. Una partición de MB es un bloque de muestras de luminancia y de dos bloques correspondientes de muestras de crominancia resultantes de una partición de un macrobloque para la interpredicción para una imagen que tenga tres matrices de muestras o un bloque de muestras de luminancia resultante de una partición de un macrobloque para la interpredicción de una imagen monocromática o para una imagen que se codifique mediante tres planos de color separados. En algunos casos, el codificador de vídeo 20 puede particionar un MB en subMB. Cada subMB es una cuarta parte de las muestras de un MB, es decir, un bloque de luminancia 8x8 y dos bloques cromáticos correspondientes, de los cuales una esquina está ubicada en una esquina del MB para una imagen que tenga tres matrices de muestra o un bloque de luminancia 8x8 de los cuales una esquina está ubicada en una esquina del MB para una imagen monocromática o una imagen que se codifique usando tres planos de color separados. Una partición de subMB es un bloque de muestras de luminancia y dos bloques correspondientes de muestras de crominancia resultantes de una partición de un subMB para una interpredicción para una imagen que tenga tres matrices de muestra o un bloque de muestras de luminancia resultante de una partición de un subMB para una interpredicción para una imagen monocromática o una imagen que se codifique usando tres planos de color separados. Los bloques de luminancia y crominancia de un MB o de una partición de MB se pueden denominar genéricamente como los bloques de muestra del MB o de la partición de MB.

[0042] En la HEVC, el codificador de vídeo 20 puede generar un conjunto de unidades de árbol de codificación (CTU). Cada una de las CTU puede ser un bloque de árbol de codificación de muestras de luminancia, dos bloques de árbol de codificación de muestras de crominancia correspondientes y estructuras de sintaxis usadas para codificar las muestras de los bloques de árbol de codificación. Un bloque de árbol de codificación puede ser un bloque de muestras de tamaño NxN. Una CTU también puede denominarse "unidad de codificación más grande" (LCU). En la HEVC, un fragmento puede comprender un número entero de CTU. Las CTU de la HEVC pueden ser análogas en gran medida a los macrobloques de otras normas, tales como H.264/AVC. Sin embargo, una CTU no está limitada necesariamente a un tamaño particular y puede incluir una o más unidades de codificación (CU). Una CU puede ser un bloque de codificación de muestras de luminancia y dos bloques de codificación correspondientes de muestras de crominancia de una imagen que tenga una matriz de muestras de luminancia, una matriz de muestras de Cb y una matriz de muestras de Cr y estructuras de sintaxis usadas para codificar las muestras de los bloques de codificación. Un bloque de codificación es un bloque NxN de muestras.

[0043] Además, en la HEVC, una CU puede tener una o más unidades de predicción (PU). Una PU puede ser un bloque de predicción de muestras de luminancia, dos bloques predictivos correspondientes de muestras de crominancia de una imagen y estructuras de sintaxis usadas para predecir las muestras de bloques predictivos. Un bloque de predicción puede ser un bloque rectangular de muestras (por ejemplo, MxN) en el que se aplique la misma predicción. Un bloque de predicción de una PU de una CU puede ser una partición de un bloque de codificación de la CU. Los bloques de luminancia y crominancia de una PU pueden denominarse genéricamente bloques de muestra de la PU.

[0044] Cuando el codificador de vídeo 20 codifique una unidad de vídeo actual (tal como un MB, una partición de MB, una PU, etc.), el codificador de vídeo 20 puede generar bloques predictivos de luminancia y crominancia de la unidad de vídeo actual. El codificador de vídeo 20 puede realizar intrapredicción o interpredicción para generar los bloques predictivos. Cuando el codificador de vídeo 20 realice la intrapredicción, el codificador de vídeo 20 puede generar,

basándose al menos en parte en muestras dentro de la misma imagen que la unidad de vídeo actual, bloques predictivos de luminancia y crominancia para la unidad de vídeo actual.

[0045] Cuando el codificador de vídeo 20 realice la interpredicción para generar bloques predictivos de luminancia y de crominancia de la unidad de vídeo actual, el codificador de vídeo 20 puede generar los bloques predictivos basados en los bloques de referencia dentro de una o más imágenes de referencia. Las imágenes de referencia pueden ser imágenes distintas de la imagen que contenga el bloque de vídeo actual. Más específicamente, el codificador de vídeo 20 puede generar una primera lista de imágenes de referencia (RefPicListO) y una segunda lista de imágenes de referencia (RefPicList1). RefPicListO y RefPicList1 son listas de imágenes de referencia. Si el codificador de vídeo 20 usa una interpredicción unidireccional para codificar la unidad de vídeo actual, el codificador de vídeo 20 puede indicar un índice de referencia que indique una posición dentro de RefPicListO o de RefPicList1 de una imagen de referencia que incluya un bloque de referencia. El codificador de vídeo 20 también puede indicar un vector de movimiento que indique un desplazamiento espacial entre el bloque de luminancia de la unidad de vídeo actual y el bloque de referencia. Si el codificador de vídeo 20 usa una interpredicción bidireccional, el codificador de vídeo 20 puede señalar dos índices de referencia que indiquen posiciones dentro de RefPicListO y de RefPicList1 de imágenes de referencia que contengan bloques de referencia. El codificador de vídeo 20 también puede indicar vectores de movimiento que indiquen desplazamientos espaciales entre el bloque de luminancia de la unidad de vídeo actual y los bloques de referencia.

[0046] En H.264/AVC, cada interMB (es decir, cada MB que se codifique usando interpredicción) se puede particionar en una de cuatro formas diferentes: una partición de MB 16x16, dos particiones de MB 16x8, dos particiones MB 8x16 o cuatro particiones de MB 8x8. Diferentes particiones de MB en un bloque pueden tener diferentes índices de referencia para cada dirección (RefPicListO o RefPicList1). Cuando un MB no está particionado en cuatro particiones de MB 8x8, el MB tiene sólo un vector de movimiento para toda la partición de MB en cada dirección. En este caso, una partición de MB puede tener un tamaño de 16x16, 8x16 o 16x8. Cuando un MB se particione en cuatro particiones de MB 8x8, cada partición de MB 8x8 puede particionarse además en subbloques, cada uno de los cuales puede tener un vector de movimiento diferente en cada dirección. Hay cuatro formas diferentes de particionar una partición de MB 8x8 en subbloques: un subbloque 8x8, dos subbloques 8x4, dos subbloques 4x8 y cuatro subbloques 4x4. Cada uno de los subbloques puede tener un vector de movimiento diferente en cada dirección.

[0047] En H.264/AVC, el codificador de vídeo 20 puede indicar un vector de movimiento para una unidad de vídeo actual (por ejemplo, un MB o una partición de MB) mediante la señalización de una diferencia de vector de movimiento (MVD). Una MVD indica una diferencia entre un predictor de vector de movimiento y un vector de movimiento para la unidad de vídeo actual. El predictor de vector de movimiento puede ser un vector de movimiento de un bloque vecino. El bloque vecino puede estar encima o a la izquierda de los bloques de muestra de la unidad de vídeo actual. Si un bloque vecino no está disponible para su uso en la generación del predictor de vector de movimiento, las componentes horizontales y verticales del predictor de vector de movimiento pueden ser iguales a 0. Entre otras razones, un bloque vecino puede no estar disponible si el bloque vecino y el bloque actual están en diferentes sectores, el bloque vecino no está dentro del límite de la imagen actual, y así sucesivamente.

[0048] Además, en H.264/AVC, el codificador de vídeo 20 puede generar una estructura de sintaxis de capa de MB que incluya elementos de sintaxis para un MB. La estructura de sintaxis de capa de MB puede incluir una estructura de sintaxis de predicción de MB o una estructura de sintaxis de predicción de subMB, dependiendo de un modo de partición del MB. La estructura de sintaxis de predicción de MB o la estructura de sintaxis de predicción de subMB pueden incluir elementos de sintaxis que indiquen la información de movimiento para el MB o la información de movimiento para las particiones de MB del MB. Por ejemplo, las estructuras de sintaxis de predicción de subMB y las estructuras de sintaxis de predicción de MB pueden incluir elementos de sintaxis que especifiquen índices de referencia y MVD.

[0049] Además, en la HEVC, cuando el codificador de vídeo 20 use la interpredicción para generar los bloques predictivos para una PU actual, el codificador de vídeo 20 puede señalar la información de movimiento para la PU actual usando un modo de combinación o un modo de predicción adaptativa de vector de movimiento (AMVP). En el modo de combinación o en el modo de AMVP, el codificador de vídeo 20 puede generar una lista de candidatos de predictores (es decir, una lista de candidatos). Los candidatos del predictor pueden especificar la información de movimiento de las PU distintas de la PU actual. En el modo de combinación, el codificador de vídeo 20 puede indicar una posición dentro de la lista de candidatos de un candidato de predictor seleccionado. La información de movimiento de la PU puede ser la misma que la información de movimiento especificada por el candidato de predictor seleccionado. En el modo de AMVP, el codificador de vídeo 20 puede indicar una posición dentro de la lista de candidatos de un candidato de predictor seleccionado, un índice de referencia y una MVD para la PU actual. La MVD para la PU actual puede basarse en una diferencia entre un vector de movimiento del candidato de predictor seleccionado y un vector de movimiento de la PU actual.

[0050] Después de que el codificador de vídeo 20 genere un bloque predictivo que corresponda a una unidad de vídeo actual (tal como un MB, una partición de MB, una PU, etc.), el codificador de vídeo 20 puede generar un bloque residual. Cada muestra en el bloque residual puede basarse en una diferencia entre las muestras correspondientes en un bloque de muestra de la unidad de vídeo actual y el bloque predictivo. El codificador de vídeo 20 puede aplicar una

transformada al bloque residual para generar uno o más bloques de coeficientes de transformada. El codificador de vídeo 20 puede cuantificar los bloques de coeficientes de transformada para reducir más el número de bits usados para representar la unidad de vídeo actual. Después de cuantificar un bloque de coeficiente de transformada, el codificador de vídeo 20 puede codificar por entropía los elementos de sintaxis que representen los coeficientes de transformada en el bloque de coeficiente de transformada y otros elementos de sintaxis. Por ejemplo, el codificador de vídeo 20 puede realizar codificación aritmética binaria adaptativa al contexto (CABAC), codificación de longitud variable adaptativa al contexto (CAVLC), codificación de Golomb exponencial u otro tipo de codificación por entropía en los elementos de sintaxis. El codificador de vídeo 20 puede emitir un flujo de bits que incluya los elementos de sintaxis codificados por entropía.

[0051] Para aplicar la codificación CABAC a un elemento de sintaxis, el codificador de vídeo 20 puede binarizar el elemento de sintaxis para formar una serie de uno o más bits, que se denominan "bins". Además, el codificador de vídeo 20 puede identificar un contexto de codificación. El contexto de codificación puede identificar probabilidades de codificación de bins que tengan valores particulares. Por ejemplo, un contexto de codificación puede indicar una probabilidad de 0,7 de codificar un bin de valor 0 y una probabilidad de 0,3 de codificar un bin de valor 1. Después de identificar el contexto de codificación, el codificador de vídeo 20 puede dividir un intervalo en un subintervalo inferior y en un subintervalo superior. Uno de los subintervalos puede estar asociado con el valor 0 y el otro subintervalo puede estar asociado con el valor 1. El ancho de los subintervalos puede ser proporcional a las probabilidades indicadas para los valores asociados por el contexto de codificación identificado. Si un bin del elemento de sintaxis tiene el valor asociado con el subintervalo inferior, el valor codificado puede ser igual al límite inferior del subintervalo inferior. Si el mismo bin del elemento de sintaxis tiene el valor asociado con el subintervalo superior, el valor codificado puede ser igual al límite inferior del subintervalo superior. Para codificar el siguiente bin del elemento de sintaxis, el codificador de vídeo 20 puede repetir estas etapas con el intervalo estando el subintervalo asociado con el valor del bit codificado. Cuando el codificador de vídeo 20 repita estas etapas para el siguiente bin, el codificador de vídeo 20 puede usar probabilidades modificadas basadas en las probabilidades indicadas por el contexto de codificación identificado y en los valores reales de los contenedores codificados.

[0052] El codificador de vídeo 20 puede emitir un flujo de bits que incluya los elementos de sintaxis codificados por entropía. El flujo de bits puede incluir una secuencia de bits que forme una representación de imágenes codificadas y datos asociados. El flujo de bits puede comprender una secuencia de unidades de capa de abstracción de red (NAL). Cada una de las unidades NAL incluye una cabecera de unidad NAL y encapsula una carga útil de secuencia de octetos sin procesar (RBSP). La cabecera de la unidad NAL puede incluir un elemento de sintaxis que indique un código de tipo de unidad NAL. El código de tipo de unidad NAL especificado por la cabecera de unidad NAL de una unidad NAL indica el tipo de la unidad NAL. Una RBSP puede ser una estructura de sintaxis que contenga un número entero de bytes que se encapsule dentro de una unidad NAL. En algunos casos, una RBSP incluye cero bits.

[0053] Diferentes tipos de unidades NAL pueden encapsular diferentes tipos de RBSP. Por ejemplo, un primer tipo de unidad NAL puede encapsular una RBSP para un conjunto de parámetros de imagen (PPS), un segundo tipo de unidad NAL puede encapsular una RBSP para un fragmento codificado, un tercer tipo de unidad NAL puede encapsular una RBSP para información de realce complementaria (SEI), y así sucesivamente. Las unidades NAL que encapsulan las RBSP para datos de codificación de vídeo (a diferencia de las RBSP para conjuntos de parámetros y mensajes SEI) pueden denominarse unidades NAL de la capa de codificación de vídeo (VCL).

[0054] El decodificador de vídeo 30 puede recibir un flujo de bits que incluya una representación codificada de datos de vídeo. El decodificador de vídeo 30 puede analizar el flujo de bits para decodificar elementos de sintaxis del flujo de bits. Como parte de la decodificación de los elementos de sintaxis del flujo de bits, el decodificador de vídeo 30 puede decodificar por entropía elementos de sintaxis del flujo de bits. Por ejemplo, el decodificador de vídeo 30 puede realizar la decodificación CABAC en al menos algunos elementos de sintaxis. El decodificador de vídeo 30 puede ejecutar, basándose al menos en parte en los elementos de sintaxis asociados con una unidad de vídeo actual (tal como un MB, una partición de MB, una PU, etc.), la interpredicción o la intrapredicción para generar bloques predictivos para la unidad de vídeo actual. Además, el decodificador de vídeo 30 puede cuantificar inversamente los coeficientes de transformada de los bloques de coeficientes de transformada asociados con la unidad de vídeo actual y puede aplicar una o más transformadas inversas a los bloques de coeficientes de transformada para generar un bloque residual. El decodificador de vídeo 30 puede reconstruir un bloque de muestra para el bloque de unidad de vídeo actual basado al menos en parte en el bloque residual y en el bloque de predicción. De esta manera, al reconstruir bloques de una imagen, el decodificador de vídeo 30 puede reconstruir la imagen.

[0055] Cuando decodificador de vídeo 30 realice la decodificación CABAC en un elemento de sintaxis, el decodificador de vídeo 30 puede identificar un contexto de codificación. El decodificador de vídeo 30 puede dividir entonces un intervalo en un subintervalo inferior y un subintervalo superior. Uno de los subintervalos puede estar asociado con el valor 0 y el otro subintervalo puede estar asociado con el valor 1. El ancho de los subintervalos puede ser proporcional a las probabilidades indicadas para los valores asociados por el contexto de codificación identificado. Si el valor codificado está dentro del subintervalo más bajo, el decodificador de vídeo 30 puede decodificar un bin que tenga el valor asociado con el subintervalo inferior. Si el valor codificado está dentro del subintervalo superior, el decodificador de vídeo 30 puede decodificar un bin que tenga el valor asociado con el subintervalo superior. Para decodificar un siguiente bin del elemento de sintaxis, el decodificador de vídeo 30 puede repetir estas etapas siendo el intervalo el

subintervalo que contenga el valor codificado. Cuando el decodificador de vídeo 30 repita estas etapas para el siguiente bin, el decodificador de vídeo 30 puede usar probabilidades modificadas basadas en las probabilidades indicadas por el contexto de codificación identificado y en los bins decodificados. El decodificador de vídeo 30 puede debinarizar entonces los bins para recuperar el elemento de sintaxis.

[0056] Como se mencionó anteriormente, la Codificación de Vídeo de Vistas Múltiples (MVC) es una extensión de la norma H.264/AVC. En la extensión de MVC a H.264, puede haber vistas múltiples de la misma escena desde diferentes puntos de vista. El término "unidad de acceso" se usa para referirse al conjunto de imágenes que corresponden a la misma instancia de tiempo. Por tanto, los datos de vídeo pueden conceptualizarse como una serie de unidades de acceso que se producen a lo largo del tiempo. Un "componente de vista" puede ser una representación codificada de una visualización en una única unidad de acceso. En esta divulgación, una "vista" puede referirse a una secuencia de componentes de vistas asociadas con el mismo identificador de vista.

[0057] La extensión de MVC de H.264/AVC da soporte a la predicción inter-vistas. La predicción inter-vistas es similar a la interpredicción usada en H.264/AVC y puede usar los mismos elementos de sintaxis. Sin embargo, cuando un codificador de vídeo realice una predicción inter-vistas en una unidad de vídeo actual (tal como un MB o una partición de MB), el codificador de vídeo 20 puede usar, como imagen de referencia, una imagen en la misma unidad de acceso que la unidad de vídeo actual, pero en una vista diferente. En contraste, la interpredicción convencional solo usa imágenes en diferentes unidades de acceso como imágenes de referencia.

[0058] En la MVC, una vista puede denominarse "vista base" si un decodificador de vídeo (por ejemplo, el decodificador de vídeo 30) puede decodificar imágenes en la vista sin referencia a imágenes en cualquier otro punto de vista. Cuando se codifique una imagen en una de las vistas no básicas, un codificador de vídeo (tal como el codificador de vídeo 20 o el decodificador de vídeo 30) puede añadir una imagen a una lista de imágenes de referencia si la imagen está en una vista diferente pero en la misma instancia de tiempo (es decir, una unidad de acceso) que la imagen que el codificador de vídeo esté codificando actualmente. Al igual que otras imágenes de referencia de interpredicción, el codificador de vídeo puede insertar una imagen de referencia de predicción inter-vistas en cualquier posición de una lista de imágenes de referencia.

[0059] En la MVC, la predicción inter-vistas puede recibir soporte de la compensación de movimiento de disparidad. La compensación de movimiento de disparidad usa la sintaxis de la compensación de movimiento H.264/AVC, pero puede permitir que una imagen en una vista diferente se use como imagen de referencia. La codificación de dos o más vistas puede recibir soporte de la MVC. Una de las ventajas de la MVC es que un codificador MVC puede usar más de dos vistas como una entrada de vídeo en 3D y un decodificador MVC puede decodificar una representación de vistas múltiples. Como resultado, los decodificadores de vídeo que soportan la MVC pueden procesar contenido de vídeo en 3D con más de dos vistas.

[0060] Además, hay una extensión emergente de 3DV basada en MVC a H.264/AVC. 3DV basada en MVC está diseñada para permitir mejoras en 3D manteniendo la compatibilidad de la MVC. 3DV basada en MVC proporciona mapas de profundidad. En consecuencia, 3DV basada en MVC también se puede denominar "MVC más profundidad", "MVC + D" o "extensión compatible con MVC, incluyendo la profundidad". Suzuki et al., "WD of MVC extensions for inclusion of depth maps" ["WD de extensiones de MVC para la inclusión de mapas de profundidad"], ISO/IEC/JTC1/SC29/WG11/N12351, diciembre de 2011 incorporado en el presente documento como referencia, es un borrador de 3DV compatible con MVC. Suzuki et al., "WD od MVC extensions for inclusion of depth maps" ["WD de extensiones de MVC para la inclusión de mapas de profundidad"], ISO/IEC/JTC1/SC29/WG11/N12544, febrero de 2012 es un borrador posterior de 3DV compatible con MVC.

[0061] Los mapas de profundidad son imágenes cuyos valores de píxel (por ejemplo, muestra) representan las profundidades tridimensionales de los objetos mostrados en las correspondientes imágenes de "textura". En algunos ejemplos, los valores de píxeles más brillantes en un mapa de profundidad pueden corresponder a objetos que estén más cerca de una cámara y los valores de píxeles más oscuros en un mapa de profundidad pueden corresponder a objetos que estén más alejados de la cámara. Las imágenes de "textura" pueden ser imágenes H.264/AVC normales.

[0062] En esta divulgación, la parte de textura de una vista puede denominarse "vista de textura" y la parte de profundidad de una vista puede denominarse "vista de profundidad". La parte de textura de una vista en una unidad de acceso, es decir, una vista de textura en una unidad de acceso, se puede denominar "componente de vista de textura". La parte de profundidad de una vista en una unidad de acceso, es decir, una vista de profundidad en una unidad de acceso, se puede denominar "componente de vista de profundidad". Por tanto, el término "componente de vista" puede referirse a una vista en una unidad de acceso y colectivamente tanto al componente de vista de textura como al componente de vista de profundidad de la misma unidad de acceso.

[0063] Como se ha mencionado anteriormente, existe un esfuerzo continuo para generar una extensión de 3DV a H.264/AVC, concretamente 3DV basada en AVC. Al igual que 3DV basada en MVC, 3DV basada en AVC proporciona mapas de profundidad. En 3DV basada en AVC, el codificador de vídeo 20 puede codificar un mapa de profundidad de la misma manera que otras vistas de una unidad de acceso. A diferencia de 3DV basada en MVC, 3DV basada en AVC puede permitir codificar un componente de vista de profundidad basado en un componente de vista de textura.

Esto puede aumentar la eficiencia de la codificación, pero puede aumentar la complejidad. 3DV basada en AVC puede no ser compatible con la MVC.

[0064] En 3DV basada en AVC, el codificador de vídeo 20 puede generar, basándose en los componentes de textura y de vista de profundidad disponibles, un componente de vista de textura sintética. Es decir, la predicción de síntesis de visión en bucle (VSP) recibe soporte de 3DV basada en AVC (y otras normas de codificación de vídeo) para una codificación de textura mejorada. Un componente de visualización de textura sintética puede ser un componente de vista de textura que se sintetice basándose en un mapa de profundidad y uno o más componentes de visualización de textura. Es decir, para permitir que la VSP codifique una vista actual, los componentes de la vista de profundidad y de textura previamente codificados de la misma unidad de acceso pueden usarse para la síntesis de visión.

[0065] Por ejemplo, un componente de vista de textura particular puede ser un componente de vista de textura de ojo izquierdo, y un codificador de vídeo 20 puede generar un componente de vista de textura de ojo derecho para la reproducción de 3DV. En algunos casos, se puede usar un componente de vista de textura sintética como imagen de referencia para la predicción de la unidad inter-accesos o la predicción inter-vistas. En consecuencia, una imagen sintetizada resultante de la VSP puede incluirse en las listas de imágenes de referencia iniciales (es decir, RefPicList 0 y/o RefPicList 1) después de las tramas de referencia temporales e inter-vistas. Los componentes de vista de textura sintética que se usen como imágenes de referencia se pueden denominar imágenes de referencia de síntesis de visión (VSRP), imágenes de referencia de síntesis de visión (VSP) o simplemente imágenes VSP.

[0066] En algunos modelos de prueba para 3DV basada en AVC, la VSP se realiza mediante la adición de la imagen sintetizada en una lista de imágenes de referencia, tales como RefPicListO o RefPicList1. Hay varios problemas potenciales con este enfoque. Por ejemplo, los vectores de movimiento a las imágenes de referencia VSP son típicamente muy cercanos a cero. Es decir, los vectores de movimiento para los bloques de referencia dentro de las imágenes VSP casi siempre tienen magnitudes de cero. Sin embargo, el enfoque de dichos modelos de prueba puede hacer que el contexto de las diferencias de vectores de movimiento sea menos eficiente. Por ejemplo, el vector de movimiento para un bloque VSP es típicamente 0, sin embargo, si los bloques vecinos se predicen con imágenes temporales, la predicción de vector de movimiento puede ser cercana a 0, por lo tanto, puede ser necesario que una diferencia de vector de movimiento innecesario se señale o, de lo contrario, el predictor de vector de movimiento no es suficiente.

[0067] De acuerdo con algunas de las técnicas de esta divulgación, el codificador de vídeo 20 puede señalar, en un flujo de bits, un elemento de sintaxis que indique si una unidad de vídeo actual se predice a partir de una imagen VSP. En algunos ejemplos, la unidad de vídeo actual puede ser un MB, una partición de MB u otro tipo de unidad. En ejemplos donde se usa H.264/AVC, la señalización del modo de VSP se introduce en el nivel de MB o de partición de MB para indicar si se predice un MB o una partición de MB a partir de una imagen VSP. El modo de VSP de una unidad de vídeo (por ejemplo, un MB o una partición de MB) indica si la unidad de vídeo se predice a partir de una imagen VSP.

[0068] En algunos ejemplos, cuando la unidad de vídeo actual se predice a partir de una imagen VSP, el codificador de vídeo 20 no es señal de información de movimiento para la unidad de vídeo actual. En los ejemplos donde se use H.264/AVC, la información de movimiento para la unidad de vídeo actual puede incluir uno o más índices de referencia y una o más MVD. En ejemplos en los que se use la HEVC, la información de movimiento para la unidad de vídeo actual puede incluir uno o más índices de referencia, uno o más índices candidatos de vector de movimiento, una o más MVD y un indicador de dirección de predicción.

[0069] Por otra parte, cuando la unidad de vídeo actual se predice a partir de una imagen VSP, el codificador de vídeo 20 puede generar, basándose al menos en parte en un bloque de la imagen VSP que está situado con los bloques de muestras de la unidad de vídeo actual, un bloque predictivo para la unidad de vídeo actual. El codificador de vídeo 20 puede generar un bloque residual para la unidad de vídeo actual. El bloque residual puede indicar diferencias entre un bloque de muestra de la unidad de vídeo actual y el bloque predictivo para la unidad de vídeo actual. El codificador de vídeo 20 puede transformar, cuantificar y codificar por entropía muestras del bloque residual.

[0070] El decodificador de vídeo 30 puede decodificar, a partir del flujo de bits, el elemento de sintaxis y determinar, basándose al menos en parte en el elemento de sintaxis, si la unidad de vídeo actual se predice a partir de una imagen VSP. Cuando se prediga la unidad de vídeo actual a partir de una imagen VSP, el decodificador de vídeo 30 puede generar, basándose al menos en parte en un bloque de la imagen VSP que esté situado con la unidad de vídeo actual, un bloque predictivo para la unidad de vídeo actual. El decodificador de vídeo 30 puede generar el bloque predictivo para la unidad de vídeo actual sin decodificar, a partir del flujo de bits, la información de movimiento para la unidad de vídeo actual.

[0071] Por tanto, de acuerdo con las técnicas de la presente divulgación, un codificador de vídeo puede generar un bloque predictivo para una unidad de vídeo actual (por ejemplo, un MB o una partición de MB) de tal manera que el bloque predictivo coincida con un bloque de ubicación conjunta en una imagen VSP. En otras palabras, cuando la unidad de vídeo actual se predice a partir de una imagen VSP, el codificador de vídeo copia un bloque situado conjunto de la unidad de vídeo actual de la imagen VSP.

[0072] Debido a que el decodificador de vídeo 30 puede ser capaz de generar un bloque predictivo de la unidad de vídeo actual sin decodificar la información de movimiento para la unidad de vídeo actual cuando la unidad de vídeo actual se prediga a partir de la imagen VSP, puede ser necesario incluir la imagen VSP en una lista de imágenes de referencia. Por lo tanto, de acuerdo con las técnicas de esta divulgación, en lugar de añadir una imagen VSP a una lista de imágenes de referencia, un codificador de vídeo (tal como el codificador de vídeo 20 o el decodificador de vídeo 30) no añade la imagen VSP a la lista de imágenes de referencia.

[0073] Además, de acuerdo con las técnicas de la esta divulgación, un codificador de vídeo puede seleccionar, basándose al menos en parte en la información asociada con un bloque vecino (por ejemplo, un MB vecino o una partición de MB), un contexto de codificación para la información de movimiento de codificación por entropía de una unidad de vídeo actual (por ejemplo, un MB o una partición de MB actual). Cuando se predice un bloque vecino (por ejemplo, un MB o una partición de MB vecino) a partir de una imagen VSP, el codificador de vídeo puede determinar que la información asociada con el bloque vecino no está disponible para seleccionar un contexto de codificación por entropía que codifique la información de movimiento de la unidad de vídeo actual. Por ejemplo, al construir un contexto de codificación por entropía para codificar por entropía la información de movimiento de un MB particular o de una partición de MB particular, un MB o una partición de MB que usa una imagen VSP se considera no disponible para usar en la selección del contexto de codificación por entropía. Cuando el codificador de vídeo determina que la información asociada con un bloque vecino no está disponible para usar en la selección de un contexto de codificación, el codificador de vídeo no usa información asociada con el bloque vecino para seleccionar el contexto de codificación. Por lo tanto, un bloque (por ejemplo, un MB o una partición de MB) que use una imagen VSP puede no tener impacto en el contexto de codificación por entropía para los elementos de sintaxis relacionados con el movimiento de la unidad de vídeo actual.

[0074] Como se indicó anteriormente, el codificador de vídeo 20 puede señalar los elementos de sintaxis que indiquen si las unidades de vídeo, tales como los MB, las particiones de MB, las particiones de subMB, etc., se predicen a partir de una imagen VSP. Por ejemplo, el codificador de vídeo 20 puede generar un elemento de sintaxis que indique si se predice un MB desde una imagen VSP, el codificador 20 puede generar un elemento de sintaxis que indique si una partición de MB se predice a partir de una imagen VSP y puede generar un elemento de sintaxis que indique si una partición de subMB se predice a partir de una imagen VSP. De acuerdo con las técnicas de esta divulgación, un codificador de vídeo puede usar el mismo o diferentes contextos de codificación cuando se codifiquen por entropía los elementos de sintaxis que indiquen si los MB, las particiones de MB y las particiones de subMB se predicen a partir de imágenes VSP. Por ejemplo, los elementos de sintaxis de nivel de MB o de partición de MB introducidos que indiquen el modo de VSP pueden compartir el mismo o diferentes contextos para la codificación por entropía.

[0075] El decodificador de vídeo 30 puede predecir (es decir, determinar), basándose en un predictor de vector de movimiento y en una MVD, un vector de movimiento para una unidad de vídeo actual. En H.264/AVC, el predictor de vector de movimiento puede derivarse de un vector de movimiento de un bloque vecino (por ejemplo, un MB, una partición de MB o un subMB vecino) cuando el bloque vecino esté disponible. De acuerdo con las técnicas de esta divulgación, cuando un codificador de vídeo prediga un vector de movimiento para una unidad de vídeo actual, un bloque (por ejemplo, un MB o una partición de MB) que se prediga a partir de una imagen VSP puede considerarse no disponible.

[0076] En H.264/AVC y sus extensiones, el codificador de vídeo 20 puede generar una estructura de sintaxis de cabecera de fragmento para un fragmento y una estructura de sintaxis de datos de fragmento para el fragmento. Como se indicó anteriormente, un fragmento puede incluir un número entero de MB. La estructura de sintaxis de datos de fragmento para un fragmento puede incluir estructuras de sintaxis de capa de MB para los MB del fragmento. La estructura de sintaxis de capa de MB para un MB puede incluir elementos de sintaxis para el MB. En algunas versiones del modelo de prueba de 3DV basada en AVC (3D-ATM), una estructura de sintaxis de datos de fragmento para un fragmento puede incluir elementos de sintaxis mb_skip_flag para los MB del fragmento. Cuando el fragmento es un fragmento P o un fragmento SP, el elemento de sintaxis mb_skip_flag para un MB indica si el MB se codifica en el modo P_Skip. Cuando el fragmento es un fragmento B, el elemento de sintaxis mb_skip_flag para un MB indica si el MB se codifica en el modo B_Skip. Por ejemplo, si el elemento de sintaxis mb_skip_flag para un MB es igual a 1 y el fragmento es un fragmento P o SP, el decodificador de vídeo 30 puede deducir que el mb_type para el MB es P_Skip (y el tipo MB se denomina colectivamente tipo P de MB). Si el elemento de sintaxis mb_skip_flag es igual a 1 y el fragmento es un fragmento B, el decodificador de vídeo 30 puede deducir que el mb_type para el MB es B_Skip (y el tipo de MB se denomina colectivamente tipo B de MB). En este ejemplo, si mb_skip_flag para un MB es igual a 0, el MB no se omite.

[0077] Cuando un MB se codifica en modo P_Skip, el decodificador de vídeo 30 puede derivar bloques predictivos de luminancia y crominancia para los MB de tal manera que los bloques predictivos de luminancia y crominancia del MB coincidan con los bloques de luminancia y crominancia de un MB situado en una imagen de referencia. Por lo tanto, cuando un MB se codifique en modo P_Skip, una estructura de sintaxis de capa de MB para el MB puede incluir un índice de referencia que identifique una ubicación, dentro de RefPicListO o RefPicList1, de la imagen de referencia. De forma similar, cuando se codifique un MB en el modo B_Skip, el decodificador de vídeo 30 puede derivar los bloques predictivos de luminancia y crominancia del MB de los MB situados de dos imágenes de referencia. Cuando

el MB se codifique en el modo B_Skip, la estructura de sintaxis de capa de MB para el MB puede incluir índices de referencia que identifiquen ubicaciones, dentro de RefPicList 0 y RefPicList 1, de las imágenes de referencia. Cuando el MB se codifica en modo P_Skip o modo B_Skip, la estructura de sintaxis de capa de MB para el MB no necesita incluir otros elementos de sintaxis, tales como elementos de sintaxis que especifiquen información de movimiento, niveles de coeficientes de transformada, etc.

[0078] Además, en algunas versiones de la 3D-ATM, la estructura de sintaxis de datos de fragmento por un fragmento puede incluir un elemento sintáctico de omisión de VSP que indique si un MB actual se omite de una imagen VSP. En otras palabras, el elemento de sintaxis de omisión de VSP puede indicar que los bloques predictivos de luminancia y crominancia para el MB actual coinciden con los bloques de luminancia y crominancia de la imagen VSP. Cuando el elemento de sintaxis de omisión de VSP indica que el MB actual se omite de una imagen VSP, el MB actual siempre se predice unidireccionalmente desde la imagen VSP. El elemento de sintaxis de omisión de VSP y el elemento de sintaxis mb_skip_flag pueden señalarse juntos y codificarse por entropía basándose en un contexto que esté basado en los MB superior e izquierdo actuales.

[0079] El elemento de sintaxis mb_skip_flag y el elemento de sintaxis skip_from_vsp_flag puede señalarse en una forma relativamente complicada. Esta divulgación puede referirse a este problema como el problema de complejidad de señalización del modo de omisión. Además, en algunas de las técnicas descritas anteriormente, solo una imagen VSP está disponible para el componente o fragmento de toda la vista. Algunas de dichas técnicas pueden soportar solo predicciones unidireccionales desde una imagen VSP y no se admite la predicción a partir de múltiples imágenes VSP. Esta divulgación puede referirse a este problema como el problema del modo de omisión modo de VSP unidireccional. Las técnicas adicionales de esta divulgación pueden abordar estos problemas. Estas técnicas adicionales pueden o no funcionar juntas para una solución completa.

[0080] En una técnica de ejemplo para abordar el asunto de la complejidad de la señalización en modo omisión, cuando solo una imagen VSP esté disponible, el elemento de sintaxis de indicador mb_skip y el indicador que indique que se omite de una imagen VSP (por ejemplo, el elemento de sintaxis de omisión de VSP) se combinan en un único elemento de sintaxis. Este elemento de sintaxis único combinado se puede denominar en el presente documento como el elemento de sintaxis mb_skip_idc. Además, el contexto de valores del elemento de sintaxis mb_skip_idc para los bloques vecinos se puede usar para predecir el elemento de sintaxis mb_skip_idc para un MB actual.

[0081] Una primera técnica de ejemplo para abordar el asunto del modo de omisión del VSP es aplicable cuando al menos un predictor (es decir, la imagen de referencia) para un MB es una imagen VSP y solo hay una imagen VSP disponible para cada dirección de predicción. En este ejemplo, el elemento de sintaxis mb_part_vsp_flag y el elemento de sintaxis sub mb_vsp_flag se extienden a ambas direcciones con el fin de indicar si una dirección de predicción dada de una partición de MB se predice a partir de una VSP o no. Como se indicó anteriormente, el elemento de sintaxis mb_part_vsp_flag de una estructura de sintaxis de predicción de MB indica si una partición de MB actual se predice a partir de una imagen VSP. El elemento de sintaxis sub mb_vsp_flag de una estructura de sintaxis de predicción de subMB indica si una partición de MB actual se predice a partir de una imagen VSP.

[0082] Una segunda técnica de ejemplo para abordar el asunto del modo de omisión de VSP unidireccional es aplicable cuando al menos un predictor (es decir, una imagen de referencia) para un MB es una imagen VSP y solo hay una imagen VSP disponible para cada dirección de predicción. En este ejemplo, la imagen VSP permanece en una lista de imágenes de referencia (por ejemplo, RefPicListO o RefPicList1). Cuando un índice de referencia (por ejemplo, un elemento de sintaxis ref_idx) de un MB o de un subMB corresponde a la imagen VSP, la predicción bidireccional es automáticamente a partir de una imagen VSP. De forma similar a otras técnicas de esta divulgación, sin embargo, los vectores de movimiento correspondientes a dicha ref_idx (suponiendo que ref_idx pertenece a RefPicListX) no se señalan, y la información de movimiento asociada a RefPicListX de la partición de MB se considera no disponible. Esto también puede aplicarse a la predicción unidireccional.

[0083] En otra técnica de ejemplo que aborda el asunto del modo de omisión de la VSP unidireccional, múltiples imágenes VSP son compatibles. En este ejemplo, cuando cualquier indicador indica que se usa la VSP, se puede señalar un índice adicional. De forma alternativa, se proporciona una indicación directa, que toma en consideración todas las imágenes VSP posibles y la imagen de omisión normal y las señala conjuntamente con un elemento de sintaxis.

[0084] Aunque las técnicas de la presente divulgación se han descrito anteriormente en gran parte con referencia a H.264/AVC, las técnicas de la presente divulgación también pueden ser aplicables a la HEVC, y específicamente la extensión de 3DV de la HEVC. En la extensión de 3DV de la HEVC, como se propone en el documento m23783, algunas regiones de una imagen sintetizada (por ejemplo, una imagen VSP) no están disponibles porque las regiones se ocluyeron en las otras vistas (es decir, las vistas desde las cuales se sintetizó la imagen VSP). Dichas regiones de la imagen VSP se pueden denominar regiones desocluidas debido a que las regiones estaban ocultas (es decir, ocluidas) en las otras vistas. Las regiones desocluidas se identifican y marcan en un mapa binario, concretamente un mapa de disponibilidad, que controla los procesos de codificación y decodificación. El codificador de vídeo y el decodificador de vídeo pueden usar ambos el mapa de disponibilidad para determinar si una CU determinada está codificada o no. Sin embargo, las observaciones han demostrado que el rendimiento de codificación de dicha técnica

es menos que óptimo. Por lo tanto, carece de un mecanismo VSP eficiente en 3DV basada en HEVC, principalmente debido a los siguientes problemas. Primero, la síntesis de visión como un modo sólo puede ser útil para algunas regiones. Segundo, el modo de síntesis de visión no está bien integrado en todo el diseño de HEVC.

[0085] Las técnicas de la presente divulgación pueden proporcionar una solución para el soporte de VPS en 3DV basada en HEVC. De acuerdo con una o más técnicas de esta divulgación, el codificador de vídeo 20 puede señalar un indicador en un nivel de CU para indicar si una CU actual se codifica con la VSP (se predice a partir de una imagen VSP). Cuando una CU actual se codifica con la VSP (es decir, la CU actual es una CU de la VSP), el residual de la CU de la VSP se puede señalar de la misma manera que otros modos.

[0086] Además, en algunos ejemplos, el codificador de vídeo 20 puede ser una señal, para cada PU, un elemento de sintaxis (por ejemplo, una bandera) que indique si la PU se predice a partir de una VSP. En dichos ejemplos, un codificador de vídeo puede predecir (es decir, generar un bloque predictivo para) una PU en la CU con una VSP mientras que el codificador de vídeo puede predecir otra PU de la CU con otros modos, tales como Inter o Intra normal. Además, en dichos ejemplos, cuando el codificador de vídeo esté construyendo un contexto de codificación por entropía que codifique la información de movimiento de una PU, el codificador de vídeo puede usar los mismos o diferentes modelos de contexto para un indicador de nivel de CU (es decir, un elemento de sintaxis si todas las PU de una CU se predicen a partir de la imagen VSP) y un indicador de nivel de PU (es decir, un elemento de sintaxis que indique si se predice una única PU a partir de la imagen VSP).

[0087] Cuando un codificador de vídeo genera un bloque predictivo para una CU o una PU de una imagen VSP, el codificador de vídeo puede copiar el bloque situado de la CU o de la PU de la imagen VSP. En otras palabras, un bloque predictivo para una CU o una PU puede coincidir con un bloque situado de la imagen VSP.

[0088] Como se describió anteriormente, el codificador de vídeo 20 puede usar el modo de combinación o modo de AMVP para señalar la información de movimiento de una PU actual. En el modo de combinación o modo de AMVP, el codificador de vídeo 20 puede generar una lista de candidatos de predictores (es decir, una lista de candidatos). Los candidatos del predictor pueden especificar la información de movimiento de las PU distintas de la PU actual. Cuando una de las otras PU no está disponible, el codificador de vídeo 20 no incluye un candidato de predictor que especifique la información de movimiento de la otra PU. De acuerdo con las técnicas de esta divulgación, cuando un codificador de vídeo verifique la disponibilidad de una PU/CU vecina durante la predicción de vector de movimiento, el codificador de vídeo puede considerar una PU/CU que se prediga a partir de una imagen VSP (es decir, una PU/CU de VSP) que no vaya a estar disponible.

[0089] La FIG. 2 es un diagrama de bloques que ilustra un codificador de vídeo 20 de ejemplo que puede implementar las técnicas de esta divulgación. La FIG. 2 se proporciona para propósitos de explicación y no debería considerarse limitadora de las técnicas tales como las ampliamente ejemplificadas y descritas en esta divulgación. Para propósitos de explicación, esta divulgación describe principalmente el codificador de vídeo 20 en el contexto de la codificación H.264/AVC. Sin embargo, las técnicas de esta divulgación pueden ser aplicables a otras normas o procedimientos de codificación, tales como la HEVC.

[0090] En el ejemplo de la FIG. 2, el codificador de vídeo 20 incluye una unidad de procesamiento de predicción 100, una unidad de generación residual 102, una unidad de procesamiento de transformada 104, una unidad de cuantificación 106, una unidad de cuantificación inversa 108, una unidad de procesamiento de transformada inversa 110, una unidad de reconstrucción 112, una unidad de filtro 114, una memoria intermedia de imágenes decodificadas 116 y una unidad de codificación por entropía 118. La unidad de procesamiento de predicción 100 incluye una unidad de procesamiento de interpredicción 120 y una unidad de procesamiento de intrapredicción 126. La unidad de procesamiento de interpredicción 120 incluye una unidad de estimación de movimiento 122 y una unidad de compensación de movimiento 124. En otros ejemplos, el codificador de vídeo 20 puede incluir más, menos o diferentes componentes funcionales.

[0091] El codificador de vídeo 20 recibe datos de vídeo. Para codificar los datos de vídeo, el codificador de vídeo 20 puede codificar cada MB de cada imagen de los datos de vídeo. Para codificar un MB, la unidad de procesamiento de predicción 100 puede seleccionar un modo de partición para el MB. El codificador de vídeo 20 puede señalar el modo de partición para el MB que use un elemento de sintaxis mb_type en una estructura de sintaxis de capa de MB para el MB. El modo de partición para el MB puede indicar cómo los bloques de luminancia y crominancia del MB se particionan en bloques de luminancia y crominancia de particiones de MB del MB.

[0092] Un fragmento puede incluir un número entero de MB. Además, los fragmentos pueden ser fragmentos I, fragmentos P, fragmentos SP, fragmentos SI o fragmentos B. Si un MB es un fragmento I, todas las particiones de MB del MB se someten a intrapredicción. Por lo tanto, si el MB está en un fragmento I, la unidad de estimación de movimiento 122 y la unidad de compensación de movimiento 124 no realizan la interpredicción en el MB. Un fragmento SP es un fragmento que puede codificarse mediante intrapredicción o interpredicción con cuantificación de las muestras de predicción usando como máximo un vector de movimiento y un índice de referencia para predecir los valores de muestra de cada bloque. Un fragmento SP puede codificarse de tal manera que sus muestras decodificadas se puedan construir de forma idéntica a otro fragmento SP o a un fragmento SI. Un fragmento SI es un fragmento que

se codifica mediante intrapredicción sólo y mediante cuantificación de las muestras de predicción. Un fragmento SI puede codificarse de tal manera que sus muestras decodificadas se puedan construir de forma idéntica a un fragmento SP.

[0093] La unidad de procesamiento de interpredicción 120 puede realizar un proceso de construcción de lista de imágenes de referencia al comienzo de la codificación de cada fragmento P, SP o B. Si la unidad de procesamiento de interpredicción 120 está codificando un fragmento P o SP, la unidad de procesamiento de interpredicción 120 puede generar una primera lista de imágenes de referencia (por ejemplo, RefPicList0). Si la unidad de proceso de interpredicción 120 codifica un fragmento B, la unidad de procesamiento de predicción 120 puede generar la primera lista de imágenes de referencia (por ejemplo, RefPicList0) y también generar una segunda lista de imágenes de referencia (por ejemplo, RefPicList1).

[0094] Si el codificador de vídeo 20 es la codificación de una unidad de vídeo actual (por ejemplo, un MB o una partición de MB) en un fragmento P, la unidad de estimación de movimiento 122 puede buscar las imágenes de referencia en una lista de imágenes de referencia (por ejemplo, RefPicList0) para un bloque de referencia para la unidad de vídeo actual. En ejemplos donde el codificador de vídeo 20 use 3DV compatible con MVC o 3DV compatible con AVC, la lista de imágenes de referencia puede incluir imágenes de referencia inter-vistas. En ejemplos donde el codificador de vídeo 20 use 3DV compatible con AVC, las imágenes de referencia inter-vistas en la lista de imágenes de referencia pueden incluir imágenes de referencia sintetizadas basadas en un mapa de profundidad. El bloque de referencia de la unidad de vídeo actual puede ser un bloque de muestras de luminancia y dos bloques correspondientes de muestras de crominancia que correspondan más estrechamente a los bloques de luminancia y crominancia de la unidad de vídeo actual.

[0095] La unidad de estimación de movimiento 122 puede generar un índice de referencia que indique la imagen de referencia en RefPicListO que contenga un bloque de referencia de una unidad de vídeo actual en un fragmento P y un vector de movimiento que indique un desplazamiento espacial entre un bloque de muestras de luminancia de la unidad de vídeo actual y el bloque de referencia. La información de movimiento de la unidad de vídeo actual puede incluir el índice de referencia de la unidad de vídeo actual y el vector de movimiento de la unidad de vídeo actual. La unidad de compensación de movimiento 124 puede generar los bloques predictivos para la unidad de vídeo actual basada en el bloque de referencia indicado por la información de movimiento de la unidad de vídeo actual.

[0096] Si la unidad de vídeo actual está en un fragmento B, la unidad de estimación de movimiento 122 puede ejecutar la interpredicción unidireccional o la interpredicción bidireccional para la unidad de vídeo actual. Para ejecutar una interpredicción unidireccional para la unidad de vídeo actual, la unidad de estimación de movimiento 122 puede buscar las imágenes de referencia de RefPicListO o una segunda lista de imágenes de referencia (por ejemplo, RefPicList1) para un bloque de referencia para la unidad de vídeo actual. En ejemplos donde el codificador de vídeo 20 usa MVC o 3DV, RefPicListO y/o RefPicList1 pueden incluir imágenes de referencia inter-vistas. La unidad de estimación de movimiento 122 puede generar un índice de referencia que indique una posición en RefPicListO o RefPicList1 de la imagen de referencia que contenga el bloque de referencia y un vector de movimiento que indique un desplazamiento espacial entre los bloques de muestra de la unidad de vídeo actual y el bloque de referencia. La unidad de estimación de movimiento 122 también puede generar un indicador de dirección de predicción que indique si la imagen de referencia está en RefPicListO o RefPicList1.

[0097] Para realizar la interpredicción bidireccional para una unidad de vídeo actual, la unidad de estimación de movimiento 122 puede buscar las imágenes de referencia en RefPicListO para un bloque de referencia para la unidad de vídeo actual y también puede buscar las imágenes de referencia en RefPicList1 para otro bloque de referencia para la unidad de vídeo actual. La unidad de estimación de movimiento 122 puede generar índices de imagen que indiquen posiciones en RefPicListO y RefPicList1 de las imágenes de referencia que contengan los bloques de referencia. Además, la unidad de estimación de movimiento 122 puede determinar vectores de movimiento que indiquen desplazamientos espaciales entre los bloques de referencia y el bloque de luminancia de la unidad de vídeo actual. La información de movimiento de la unidad de vídeo actual puede incluir los índices de referencia y los vectores de movimiento de la unidad de vídeo actual. La unidad de compensación de movimiento 124 puede generar los bloques predictivos para la unidad de vídeo actual basada en los bloques de referencia indicados por la información de movimiento de la unidad de vídeo actual.

[0098] La unidad de procesamiento de intrapredicción 126 puede generar datos predictivos de una unidad de vídeo actual ejecutando la intrapredicción en la unidad de vídeo actual. Los datos predictivos para la unidad de vídeo actual pueden incluir bloques predictivos para la unidad de vídeo actual y diversos elementos de sintaxis. La unidad de procesamiento de intrapredicción 126 puede realizar la intrapredicción en unidades de vídeo en los fragmentos I, los fragmentos P y los fragmentos B.

[0099] La unidad de procesamiento de predicción 100 puede seleccionar los datos predictivos para una unidad de vídeo actual entre los datos predictivos generados por la unidad de procesamiento de interpredicción 120 para la unidad de vídeo actual o los datos predictivos generados por la unidad de procesamiento de predicción 126 para la unidad de vídeo actual. En algunos ejemplos, la unidad de procesamiento de predicción 100 selecciona los datos

predictivos para la unidad de vídeo actual basándose en las mediciones de velocidad/distorsión de los conjuntos de datos predictivos.

[0100] La unidad de generación residual 102 puede generar bloques residuales restando las muestras en bloques predictivos para la unidad de vídeo actual de las muestras correspondientes de los bloques de muestra de la unidad de vídeo actual. La unidad de procesamiento de transformada 104 puede generar bloques de coeficientes de transformada para cada bloque residual aplicando una o más transformadas al bloque residual. La unidad de procesamiento de transformada 104 puede aplicar diversas transformadas a un bloque residual. Por ejemplo, la unidad de procesamiento de transformada 104 puede aplicar una transformada de coseno discreta (DCT), una transformada direccional, una transformada entera, una transformada de ondículas, o una transformada conceptualmente similar a un bloque residual.

[0101] La unidad de cuantificación 106 puede cuantificar los coeficientes de transformada en un bloque de coeficiente de transformada. El proceso de cuantificación puede reducir la profundidad de bits asociada a algunos o a la totalidad de los coeficientes de transformada. Por ejemplo, un coeficiente de transformada de *n* bits puede redondearse hacia abajo a un coeficiente de transformada de *m* bits durante la cuantificación, donde *n* es mayor que *m*. La unidad de cuantificación 106 puede cuantificar un bloque de coeficiente de transformada basado en un valor de parámetro de cuantificación (QP). El codificador de vídeo 20 puede ajustar el grado de cuantificación aplicado a los bloques de coeficientes de transformada ajustando el valor de QP.

[0102] La unidad de cuantificación inversa 108 y la unidad de procesamiento de transformada inversa 110 pueden aplicar cuantificación inversa y transformadas inversas a un bloque de coeficientes, respectivamente, para reconstruir un bloque residual a partir del bloque de coeficientes de transformada. La unidad de reconstrucción 112 puede añadir las muestras en los bloques residuales reconstruidos a las muestras correspondientes de uno o más bloques predictivos generados por la unidad de procesamiento de predicción 100 para producir los bloques reconstruidos. La unidad de filtro 114 puede realizar una operación de desbloqueo para reducir los artefactos de bloqueo en los bloques reconstruidos. La memoria intermedia de imágenes decodificadas 116 puede almacenar los bloques de codificación reconstruidos después de que la unidad de filtro 114 realice las una o más operaciones de desbloqueo en los bloques de codificación reconstruidos. La unidad de estimación de movimiento 122 y la unidad de compensación de movimiento 124 pueden usar una imagen de referencia que contenga el bloque de vídeo reconstruido para realizar la interpredicción en las PU de las imágenes posteriores. Además, la unidad de procesamiento de intrapredicción 126 puede usar bloques reconstruidos en la memoria intermedia de imágenes decodificadas 116 para realizar la intrapredicción.

[0103] La unidad de codificación por entropía 118 puede recibir datos de otros componentes funcionales del codificador de vídeo 20. Por ejemplo, la unidad de codificación por entropía 118 puede recibir bloques de coeficientes de la unidad de cuantificación 106 y puede recibir elementos de sintaxis de la unidad de procesamiento de predicción 100. La unidad de codificación por entropía 118 puede realizar una o más operaciones de codificación por entropía en los datos para generar datos codificados por entropía. Por ejemplo, el codificador de vídeo 20 puede realizar una operación CAVLC, una operación CABAC, una operación de codificación de longitud variable a variable (V2V), una operación de codificación aritmética binaria adaptativa al contexto basada en sintaxis (SBAC), una entropía de partición de intervalo de probabilidad (PIPE), una operación de codificación Golomb Exponencial u otro tipo de operación de codificación por entropía en los datos.

[0104] La FIG. 3 es un diagrama de bloques que ilustra un decodificador de vídeo 30 de ejemplo que puede implementar las técnicas de esta divulgación. La FIG. 3 se proporciona para propósitos de explicación y no se limita a las técnicas como las ampliamente ejemplificadas y descritas en esta divulgación. Para propósitos de explicación, esta divulgación describe el decodificador de vídeo 30 en el contexto de la codificación H.264/AVC. Sin embargo, las técnicas de esta divulgación pueden ser aplicables a otras normas o procedimientos de codificación.

[0105] En el ejemplo de la FIG. 3, el decodificador de vídeo 30 incluye una unidad de decodificación por entropía 150, una unidad de procesamiento de predicción 152, una unidad de cuantificación inversa 154, una unidad de procesamiento de transformada inversa 156, una unidad de reconstrucción 158, una unidad de filtro 160 y una memoria intermedia de imágenes decodificadas 162. La unidad de procesamiento de predicción 152 incluye una unidad de compensación de movimiento 164 y una unidad de procesamiento de intrapredicción 166. En otros ejemplos, el decodificador de vídeo 30 puede incluir más, menos o diferentes componentes funcionales.

[0106] El decodificador de vídeo 30 puede recibir un flujo de bits. La unidad de decodificación por entropía 150 puede analizar el flujo de bits para extraer los elementos de sintaxis del flujo de bits. Como parte del análisis del flujo de bits, la unidad de decodificación por entropía 150 puede decodificar por entropía los elementos de sintaxis codificados por entropía en el flujo de bits. La unidad de procesamiento de predicción 152, la unidad de cuantificación inversa 154, la unidad de procesamiento de transformada inversa 156, la unidad de reconstrucción 158 y la unidad de filtro 160 pueden generar datos de vídeo decodificados (es decir, reconstruir los datos de vídeo) en base a los elementos de sintaxis decodificados del flujo de bits. Los elementos de sintaxis decodificados del flujo de bits pueden incluir elementos de sintaxis que representen bloques de coeficientes de transformada.

[0107] La unidad de cuantificación inversa 154 puede cuantificar inversamente, es decir, descuantificar, transformar bloques de coeficientes. La unidad de cuantificación inversa 154 puede usar un valor de QP para determinar un grado de cuantificación y, del mismo modo, un grado de cuantificación inversa para que se aplique la unidad de cuantificación inversa 154. Después de que la unidad de cuantificación inversa 154 cuantifique inversamente un bloque de coeficiente de transformada, la unidad de procesamiento de transformada inversa 156 puede aplicar una o más transformadas inversas al bloque de coeficiente de transformada para generar un bloque residual. Por ejemplo, la unidad de procesamiento de transformada inversa 156 puede aplicar una DCT inversa, una transformada entera inversa, una transformada de Karhunen-Loeve (KLT) inversa, una transformada de rotación inversa, una transformada direccional inversa u otra transformada inversa al bloque de coeficientes.

[0108] Si una unidad de vídeo actual se codifica usando la intrapredicción, la unidad de procesamiento de intrapredicción 166 puede realizar la intrapredicción para generar bloques predictivos para la unidad de vídeo actual. Por ejemplo, la unidad de procesamiento de intrapredicción 166 puede determinar un modo de intrapredicción para la unidad de vídeo actual basándose en elementos de sintaxis en el flujo de bits. La unidad de procesamiento de intrapredicción 166 puede usar el modo de intrapredicción para generar los bloques predictivos para la unidad de vídeo actual basada en los bloques vecinos espacialmente.

[0109] La unidad de compensación de movimiento 164 puede construir una primera lista de imágenes de referencia (RefPicListO) y una segunda lista de imágenes de referencia (RefPicList1) basándose en elementos de sintaxis decodificados a partir del flujo de bits. En ejemplos en los que el flujo de bits se codifica usando 3DV compatible con MVC o 3DV basada en AVC, RefPicList0 y/o RefPicList1 pueden incluir imágenes de referencia inter-vistas. En ejemplos donde el flujo de bits se codifique usando 3DV basada en AVC, las imágenes de referencia inter-vistas en RefPicList0 y/o RefPicList1 pueden incluir imágenes de referencia sintetizadas basadas en mapas de profundidad. Además, si la unidad de vídeo actual se codifica usando la interpredicción, la unidad de decodificación por entropía 150 puede decodificar la información de movimiento para la unidad de vídeo actual. La unidad de compensación de movimiento 164 puede determinar, basándose en la información de movimiento de la unidad de vídeo actual, uno o más bloques de referencia para la unidad de vídeo actual. La unidad de compensación de movimiento 164 puede generar, basándose en uno o más bloques de referencia para la unidad de vídeo actual, bloques predictivos para la unidad de vídeo actual.

[0110] La unidad de reconstrucción 158 puede reconstruir bloques de muestra de la unidad de vídeo actual basándose en los bloques residuales para la unidad de vídeo actual y en los bloques predictivos para la unidad de vídeo actual. En particular, la unidad de reconstrucción 158 puede añadir muestras (por ejemplo, componentes de luminancia o crominancia) de los bloques residuales a las muestras correspondientes de los bloques predictivos para reconstruir los bloques de muestra de la unidad de vídeo actual.

[0111] La unidad de filtro 160 puede ejecutar una operación de desbloqueo para reducir los artefactos de bloqueo asociados con los bloques de muestra de la unidad de vídeo actual. La unidad de filtro 114 del codificador de vídeo 20 puede realizar una operación de desbloqueo similar a la operación de desbloqueo de la unidad de filtro 160, por lo tanto, por concisión, esta divulgación solo describe la operación de desbloqueo con respecto a la unidad de filtro 160. Cuando la unidad de filtro 160 realice la operación de desbloqueo en H.264/AVC, la unidad de filtro 160 puede realizar un proceso de filtrado para los bordes del bloque. La unidad de filtro 160 puede aplicar el proceso de filtrado a un conjunto de ocho muestras a través de un borde horizontal o vertical del bloque 4x4. Estas muestras pueden denominarse "muestras de entrada" y se pueden indicar con pi y qi con i = 0,3 con el borde entre p_0 y q_0. Cuando la unidad de filtro 160 aplique el proceso de filtrado al conjunto de muestras, la unidad de filtro 160 puede determinar un valor de resistencia de filtrado límite (bS). Además, la unidad de filtro 160 puede determinar los parámetros de cuantificación (qP_p, qP_q) para los bloques. La unidad de filtro 160 puede realizar entonces un proceso de derivación de umbral basado al menos en parte en los valores de muestra, bS, en las desviaciones de filtro, qP_p y qP_q. El proceso de derivación de umbral puede devolver un valor que indique si las muestras de entrada están filtradas. El proceso de derivación de umbral también puede devolver un valor (índice A) y valores de las variables de umbral α y β. La unidad de filtro 160 puede realizar entonces una operación, basada al menos en parte en bS, α, β y el índice A, un filtro para las muestras de entrada.

[0112] Como se mencionó anteriormente, la unidad de filtro 160 puede determinar un valor de resistencia de filtrado límite (bS). La unidad de filtro 160 puede determinar bS basándose en una variedad de diferentes tipos de información. Por ejemplo, la unidad de filtro 160 puede determinar bS basándose al menos en parte en los modos de predicción (por ejemplo, inter o intra) de los bloques, en los índices de referencia de los bloques, si los bloques se someten a interpredicción de manera unidireccional o bidireccional, en los vectores de movimiento de los bloques, y así sucesivamente.

[0113] El decodificador de vídeo 30 puede almacenar bloques reconstruidos en la memoria intermedia de imágenes decodificadas 162. La memoria intermedia de imágenes decodificadas 162 puede proporcionar imágenes de referencia para la posterior compensación de movimiento, la intrapredicción y la presentación en un dispositivo de visualización, tal como el dispositivo de visualización 32 de la FIG. 1. Por ejemplo, el decodificador de vídeo 30 puede realizar, basándose en los bloques de la memoria intermedia de imágenes decodificadas 162, operaciones de intrapredicción o de interpredicción en las PU de otras CU.

[0114] Como se analizó anteriormente, el flujo de bits puede comprender una serie de unidades NAL. Las unidades NAL pueden incluir unidades NAL de fragmento codificadas que encapsulen segmentos codificados de imágenes de los datos de vídeo. Cada fragmento codificado incluye una estructura de sintaxis de cabecera de fragmento y una estructura de sintaxis de fragmento de datos. De acuerdo con una primera implementación de ejemplo de las técnicas de esta divulgación, la estructura de sintaxis de cabecera de fragmento puede ajustarse a la sintaxis de ejemplo de la Tabla 1, a continuación.

TABLA 1: Sintaxis de cabecera de fragmento

slice_header() {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(nal_unit_type = = 5)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type = = 0) {		
pic_order_cnt_lsb	2	u(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type = = 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0]	2	se(v)
if(pic_order_present_flag && ! field_pic_flag)		
delta_pic_order_cnt[1]	2	se(v)
}		
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
if(slice_type = = B)		
direct_spatial_mv_pred_flag	2	u(1)
if (seq_vsp_enabled_flag)		
slice_vsp_flag	2	u(1)
if(slice_type = = P || slice_type = = SP || slice_type = = B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		
num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type = = B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
ref_pic_list_reordering() if((weighted_pred_flag && (slice_type = = P || slice_type = = SP)) || (weighted bipred idc = = 1 && slice type = = B))	2	

slice_header() {	C	Descriptor
pred_weight_table()	2	
if(nal_ref_idc != 0)		
dec_ref_pic_marking()	2	
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
if(slice_type = = SP || slice_type = = SI) {		
if(slice_type = = SP)		
sp_for_switch_flag	2	u(1)
slice_qs_delta	2	se(v)
}		
if(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
if(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
if(num_slice_groups_minusl > 0 && slice_group map_type >= 3 && slice group map type <= 5)		
slice_group_change_cycle	2	u(v)
}		

[0115] En el ejemplo de la Tabla 1 anterior y en otras tablas de sintaxis de esta divulgación, los elementos de sintaxis con descriptor de tipo ue(v) pueden ser números enteros de longitud variable sin signo codificados usando codificación Golomb exponencial (Exp-golomb) de orden 0 con el bit izquierdo primero. En el ejemplo de la Tabla 1 y en las siguientes tablas, los elementos de sintaxis que tienen descriptores de la forma u(n), donde *n* es un entero no negativo, son valores sin signo de longitud *n*.

[0116] La variable seq_vsp_enabled_flag de la Tabla 1 indica si se permite la VSP para una secuencia de vídeo codificada. Si seq_vsp_enabled_flag indica que la VSP se permite para una secuencia de vídeo codificada que incluya un fragmento, una estructura de sintaxis de cabecera de fragmento para el fragmento puede incluir un elemento de sintaxis slice_vsp_flag. El elemento de sintaxis slice_vsp_flag indica si se permite la VSP para el fragmento. Cuando slice_vsp_flag no está presente, el decodificador de vídeo 30 puede deducir (es decir, determinar automáticamente) que slice_vsp_flag es igual a 0, lo que significa que la VSP no está permitida para el fragmento.

[0117] Una unidad NAL que incluya una estructura de sintaxis de cabecera de fragmento también puede incluir una estructura de sintaxis de datos de fragmento. Una estructura de sintaxis de cabecera de fragmento corresponde a una estructura de sintaxis de datos de fragmento si la estructura de sintaxis de cabecera de fragmento y la estructura de sintaxis de datos de fragmento están encapsuladas por la misma unidad NAL, y viceversa. La estructura de sintaxis de datos de fragmento puede incluir estructuras de sintaxis de capa de MB para MB de un fragmento. La Tabla 2 siguiente es una sintaxis de ejemplo para una estructura de sintaxis de datos de fragmento de acuerdo con la implementación del primer ejemplo de las técnicas de esta divulgación.

TABLA 2 - Sintaxis de datos de fragmento

slice_data() {	C	Descriptor
if(entropy_coding_mode_flag)		
while(!byte_aligned())		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		

slice_data() {	C	Descriptor
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data()		
} else {		
mb_skip_flag	2	ae(v)
if (slice_vsp_flag && mb_skip_flag && VspRefExist)		
skip_from_vsp_flag		ae(v)
moreDataFlag = !mb_skip_flag		
}		
if(moreDataFlag) {		
if(MbaffFrameFlag && (CurrMbAddr % 2 = = 0 ‖ (CurrMbAddr % 2 = = 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) | ae(v)
macroblock_layer()	2 | 3 | 4	
}		
if(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data()		
else {		
if(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
if(MbaffFrameFlag && CurrMbAddr % 2 = = 0)		
moreDataFlag = 1		
else {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} while(moreDataFlag)		
}		

[0118] En la sintaxis de ejemplo de la Tabla 2, una estructura de sintaxis de datos de fragmento puede incluir un elemento de sintaxis mb_skip_flag. El elemento de sintaxis mb_skip_flag indica si un MB se codifica usando el modo de omisión. Todo el MB se puede predecir a partir de una imagen VSP o se puede usar el modo de omisión normal. En otras palabras, se puede generar un bloque predictivo para el MB a partir de una imagen VSP o de otra imagen. Por ejemplo, mb_skip_flag igual a 1 puede indicar que el modo de omisión se usa para codificar el MB. En este ejemplo, mb_skip_flag igual a 0 puede indicar que el modo de omisión no se usa para codificar el MB.

[0119] Además, en la sintaxis de ejemplo de la Tabla 2, una estructura de sintaxis de datos de fragmento puede incluir un skip_from_vsp_flag para un MB cuando slice_vsp_flag de la estructura de sintaxis de cabecera de fragmento correspondiente indique que la VSP se permite para el fragmento, que el MB se codifica usando el modo de omisión y que hay una imagen VSP en RefPicListO o RefPicList1. En la Tabla 2, la variable VspRefExist indica si hay una imagen VSP en RefPicListO o RefPicList1. En algunos ejemplos, se omite la condición VspRefExist. Skip_from_vsp_flag para un MB indica si todo el MB se predice desde una imagen VSP. Por ejemplo, skip_from_vsp_flag igual a 1 puede indicar que, cuando se use el modo de omisión, todo el MB se predice a partir de una imagen VSP. En este ejemplo, skip_from_vsp_flag igual a 0 puede indicar el modo de omisión normal. Cuando skip_from_vsp_flag no está presente en la estructura de sintaxis de datos de fragmento, el decodificador de vídeo 30 puede deducir que el MB se codifica usando el modo de omisión normal (por ejemplo, el decodificador de vídeo 30 puede deducir que skip_from_vsp_flag es igual a cero).

[0120] Como se muestra en la Tabla 2 anterior, los datos de fragmento pueden incluir una o más estructuras sintácticas de capa de MB. Una estructura de sintaxis de capa de MB puede incluir un MB codificado. La Tabla 3 siguiente es una sintaxis de ejemplo para la estructura de sintaxis de capa de MB de acuerdo con la implementación del primer ejemplo de las técnicas de esta divulgación.

TABLA 3 - Sintaxis de capa de macrobloque

macroblock_layer() {	**C**	**Descriptor**
if (slice_vsp_flag && VspRefExist)		
vsp_mb_flag	2	ae(v)
if (! vsp_mb_flag)		
mb_type	2	ue(v) | ae(v)
if(mb_type = = I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	3	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i**]**	3	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i **]**	3	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		

if(mb_type != I_NxN &&
MbPartPredMode(mb_type, 0) != Intra_16x16 &&
NumMbPart(mb_ type) = = 4) {		

if (!vsp_mb_flag)		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		

macroblock_layer() {	C	Descriptor
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type = = I_NxN)		
transform_size_8x8_flag	2	u(1) | ae(v)
if (!vsp_mb_flag)		
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_16x16) {		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (vsp_mb_flag || (!vsp_mb_flag && (mb_type != B_Direct_16x16 || direct 8x8 _inference_flag))))		
transform_size_8x8_flag	2	u(1) | ae(v)
}		
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual(0,15)	3 | 4	
}		
}		
}		

[0121] En la sintaxis de ejemplo de la Tabla 3, una estructura de sintaxis de capa de MB para un MB puede incluir un elemento de sintaxis vsp_mb_flag cuando se permita la VSP en un fragmento que contenga el MB y haya una imagen VSP en RefPicList0 o RefPicList1. En otras palabras, la estructura de sintaxis de capa de MB puede incluir vsp_mb_flag cuando slice_vsp_flag sea igual a 1 y VspRefExist sea igual a 1. En algunos ejemplos, se omite la condición VspRefExist. El elemento de sintaxis vsp_mb_flag puede indicar si se predice todo el MB a partir de una imagen VSP. Por ejemplo, vsp_mb_flag igual a 1 puede indicar que todo el MB se predice a partir de una imagen VSP. En este ejemplo, vsp_mb_flag igual a 0 indica que todo el MB puede predecirse por otros modos.

[0122] Cuando se predice todo el MB a partir de una imagen VSP (por ejemplo, cuando vsp_mb_flag es igual a 1), el elemento de sintaxis mb_type no se señala en la estructura de sintaxis de capa de MB. El elemento de sintaxis mb_type indica un tipo del MB. Además, cuando vsp_mb_flag no esté presente, el decodificador de vídeo 30 puede deducir vsp_mb_flag para indicar que todo el MB puede predecirse por otros modos (por ejemplo, el decodificador de vídeo 30 puede deducir que vsp_mb_flag es igual a 0).

[0123] Además, el ejemplo de la Tabla 3 incluye una variable noSubMbPartLessThan8x8Flag. Si el número de particiones de MB del MB actual es menor que cuatro, noSubMbPartLessThan8x8Flag es verdadero, como en H.264/AVC. Sin embargo, si el número de particiones de MB del MB actual es igual a cuatro (es decir, si NumMbPart (mb_type) = 4), noSubMbPartLessThan8x8Flag puede derivarse comprobando solo las particiones de MB 8x8 con mb_part_vsp_flag igual a 0. Por ejemplo, si una partición de MB tiene una partición de subMB menor que 8x8, noSubMbPartLessThan8x8Flag puede ser falso. De lo contrario, noSubMbPartLessThan8x8Flag puede ser verdadero (es decir, noSubMbPartLessThan8x8Flag = 1).

[0124] Además, como se muestra en el ejemplo de sintaxis de la Tabla 3, el codificador de vídeo 20 puede determinar, basándose al menos en parte en si todo el MB se predice a partir de una imagen VSP, si se debe incluir un elemento de sintaxis (transform_size_8x8_flag) en la estructura de sintaxis de capa de MB. Asimismo, el decodificador de vídeo 30 puede determinar, basándose al menos en parte en si todo el MB se predice a partir de una imagen VSP, si decodificar el elemento de sintaxis de la estructura de sintaxis de capa de MB. transform_size_8x8_flag puede indicar

si, para el MB actual, se invoca un proceso de decodificación de coeficientes de transformada y un proceso de construcción de imágenes previo a un proceso de filtro de desbloqueo para los bloques residuales 8x8 o 4x4 para las muestras de luminancia y, en algunos casos, para las muestras de Cb y Cr.

[0125] Como se muestra en la sintaxis de ejemplo de la Tabla 3, una estructura de sintaxis de capa de MB puede incluir una estructura de sintaxis de predicción de MB (mb_pred (mbtype)) si todo el MB no se predice a partir de una imagen VSP (por ejemplo, si vsp_mb_flag no es igual a 1). Por el contrario, la estructura de sintaxis de capa de MB no incluye una estructura de sintaxis de predicción de MB si todo el MB se predice a partir de una imagen VSP (por ejemplo, si vsp_mb_flag es igual a 1). En otros ejemplos, la estructura de sintaxis de capa de MB puede incluir la estructura de sintaxis de predicción de MB, independientemente de si el MB se predice a partir de la imagen VSP.

[0126] Como se mostró en el ejemplo de la Tabla 3, anteriormente, una estructura de sintaxis de capa de MB puede incluir una estructura de sintaxis de predicción de MB. La estructura de sintaxis de predicción de MB puede incluir información de movimiento para un MB. La Tabla 4 siguiente es una sintaxis de ejemplo para una estructura de sintaxis de predicción de MB de acuerdo con la primera implementación de ejemplo de las técnicas de esta divulgación.

TABLA 4 - Sintaxis de predicción de macrobloque

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_ type, 0) = = Intra 16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx]	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag[luma4x4BlkIdx])		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if (slice_vsp_flag && NumMbPart(mb_type) != 1 && VspRefExist) //vsp_mb_ flag is not 1		
mb_part_vsp_flag[mbPartIdx]	2	ae(v)
if(!mb_part_vsp_flag[mbPartIdx] &&(num_ref_idx_l0_active_minus1 > 0 | | mb_field_decoding_flag) && MbPartPredMode(mb_ type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx]	2	te(v) | ae(v)
for (mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type) && !mb part_vsp_flag [mbPartIdx]; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!**mb_part_vsp_flag[** mbPartIdx] && MbPartPredMode (mb type, mbPartIdx) != Pred L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx][0][compIdx]	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		

mb_pred(mb_type) {	C	Descriptor
if(!**mb_part_vsp_flag[** mbPartIdx] && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0))		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx][0][compIdx]	2	se(v) | ae(v)
}		
}		

[0127] En la sintaxis de ejemplo de la Tabla 4, una estructura de sintaxis de predicción de MB puede incluir una matriz de elementos de sintaxis mb_part_vsp_flag []. Cada entrada en mb_part_vsp_flag [] indica si se predice una partición de MB diferente de un MB a partir de una imagen VSP. La estructura de sintaxis de predicción de MB puede incluir un mb_part_vsp_flag para un particular MB si la VSP está habilitada para un fragmento que contenga la partición de MB en particular (por ejemplo, slice_vsp_flag = 1), si el número de particiones de MB del MB no es igual a 1 y si hay una imagen VSP en RefPicListO o RefPicList1 (por ejemplo, VspRefExist = 1). En otros ejemplos, la estructura de sintaxis de predicción de MB puede incluir mb_part_vsp_flag si la VSP está habilitada para un fragmento que contenga la partición de MB particular (por ejemplo, slice_vsp_flag = 1) y no evalúa si el número de particiones de MB del MB no es igual a 1 o VspRefExist = 1.

[0128] En algunos ejemplos, mb_part_vsp_flag [mbPartIdx] igual a 1 puede indicar que una partición de MB particular se predice a partir de una imagen VSP, donde mbPartIdx es un índice de la partición de MB particular. En este ejemplo, mb_part_vsp_flag [mbPartIdx] igual a 0 puede indicar que toda la partición de MB no se predice a partir de una imagen VSP, donde mbPartIdx es un índice de la partición de MB. En algunos ejemplos, cuando mb_part_vsp_flag [mbPartIdx] no está presente, el decodificador de vídeo 30 puede deducir mb_part_vsp_flag [mbPartIdx] para indicar que toda la partición de MB no se predice a partir de una imagen VSP (por ejemplo, el decodificador de vídeo 30 puede deducir que mb_part_vsp_flag [mbPartIdx] igual a 0).

[0129] Como se muestra en la Tabla 4, si mbPartIdx es el índice de una partición de MB particular, una estructura de sintaxis de predicción de MB no incluye índices de referencia (ref_idx_l0 y ref_idx_l1) o diferencias de vectores de movimiento (mvd_l0 y mvd_l1) para la partición de MB particular cuando mb_part_vsp_flag [mbPartIdx] indica que la partición de MB particular se predice a partir de una imagen VSP. En otros ejemplos, la estructura de sintaxis de predicción de MB puede incluir ref idx_l0 o ref idx_l1 independientemente de si la partición de MB particular se predice a partir de una imagen VSP.

[0130] Además, si un MB no se predice por completo a partir de una imagen VSP (por ejemplo, si el vsp_mb_flag del MB no es igual a 1), la estructura de sintaxis de capa de MB para el MB puede incluir una estructura de sintaxis de predicción de subMB. Las estructuras de sintaxis de predicción de subMB pueden incluir información de movimiento (por ejemplo, vectores de movimiento, índices de referencia, etc.) para particiones de MB. La Tabla 5 siguiente muestra una sintaxis de ejemplo para una estructura de sintaxis de predicción de subMB de acuerdo con la implementación del primer ejemplo de las técnicas de esta divulgación.

TABLA 5 - Sintaxis de predicción de submacrobloque

sub_mb_pred(mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++) {		
if (slice_vsp flag && VspRefExist) // vsp mb flag is not 1		
sub_mb_vsp_flag[mbPartIdx]		
if (!sub mb_vsp_flag[mbPartIdx])		
sub_mb_type[mbPartIdx]	2	ue(v) | ae(v)
}		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag[mbPartIdx] &&(num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub _mb_ type[mbPartIdx]) != Pred_L1)		

sub_mb_pred(mb_type) {	C	Descriptor
ref_idx_l0[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag[mbPartIdx] && (num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag[mbPartIdx] &&(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub _mb_ type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0;		
subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx][subMbPartIdx][compIdx]	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag[mbPartIdx] &&sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub _mb_ type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx][subMbPartIdx][compIdx]	2	se(v) | ae(v)
}		

[0131] En la sintaxis de ejemplo de la Tabla 5, una estructura de sintaxis de predicción de subMB puede incluir elementos de sintaxis sub_mb_vsp_flag []. Cada sub_mb_vsp_flag [] indica si se predice una partición de MB diferente (por ejemplo, 8x8) a partir de una imagen VSP. La estructura de sintaxis de predicción de subMB puede incluir un sub_mb_vsp_flag para una partición de MB particular cuando la VSP esté habilitada para un fragmento que contenga la partición de MB particular (por ejemplo, slice_vsp_flag = 1) y una imagen VSP esté en RefPicListO o RefPicList1 (por ejemplo, cuando VspRefExist = 1). En otros ejemplos, la condición de VspRefExist = 1 no está determinada.

[0132] En un ejemplo, sub_mb_vsp_flag[mbPartIdx] igual a 1 puede indicar que una partición de MB particular se prediga a partir de una imagen VSP, en la que mbPartIdx es un índice de la partición de MB particular. En este ejemplo, sub_mb_vsp_flag [mb-PartIdx] igual a 0 puede indicar que toda la partición de MB particular no se predice a partir de una imagen VSP. Cuando sub_mb_vsp_flag[mbPartIdx] no está presente, toda la partición de MB particular no se predice a partir de una imagen VSP (por ejemplo, el decodificador de vídeo 30 puede deducir sub_mb_vsp_flag[mbPartIdx] para que sea igual a 0).

[0133] Además, como se muestra en la sintaxis de ejemplo de la Tabla 5, cuando mbPartIdx es el índice de una partición de MB particular y mb_part_vsp_flag [mbPartIdx] indica que la partición de MB particular se predice a partir de una imagen VSP, la estructura de sintaxis de predicción de subMB no incluye un elemento de sintaxis que especifique un tipo de subMB de la partición de MB particular (sub_mb_type [mbPartIdx]). Además, cuando mb_part_vsp_flag [mbPartIdx] indica que la partición de MB particular se predice a partir de una imagen VSP, la estructura de predicción de subMB no incluye índices de referencia (ref_idx_l0 [mb-PartIdx] y ref idx_l1 [mbPartIdx]) para la partición de MB particular. Además, cuando mb_part_vsp_flag [mbPartIdx] indica que la partición de MB particular se predice a partir de una imagen VSP, la estructura de sintaxis de predicción de subMB no incluye las diferencias de vectores de movimiento (mvd_l0 [mbPartIdx] y mvd_l1 [mbPartIdx]) para la partición de MB particular.

[0134] Las Tablas 1-5 anteriores incluyen elementos de sintaxis vsp_mb_flag, mb_part_flag, sub_mb_vsp_flag y skip_from_vsp_flag. El decodificador de vídeo 30 puede usar los elementos de sintaxis vsp_mb_flag, mb_part_flag, sub_mb_vsp_flag y skip_from_vsp_flag para derivar un indicador, concretamente, VSPFlag, para una partición de MB

de cualquier tamaño. El VSPFlag puede indicar si se predice una partición de MB a partir de una imagen VSP como se describe en otra parte en esta divulgación. Por ejemplo, cuando VSPFlag para una partición de MB es igual a 1, la partición de MB puede predecirse a partir de la imagen VSP.

[0135] De acuerdo con la primera implementación de las técnicas de esta divulgación, cuando el VSPFlag indica que una partición de MB se predice a partir de una imagen VSP (por ejemplo, cuando el VSPFlag es igual a 1), la unidad de compensación de movimiento 164 puede establecer las muestras de los bloques predictivos de la partición de MB en las muestras de la imagen VSP. Por ejemplo, supongamos que la partición de MB actual tiene un píxel superior izquierdo con coordenadas (x, y) en el componente de luminancia y (x ', y') en los componentes de crominancia. Además, supongamos que el tamaño de la partición de MB actual es NxM, donde N, M es igual a 8 o 16. En esta instancia, la unidad de compensación de movimiento 164 puede, partiendo de las coordenadas (x, y) con el mismo tamaño de NxM, establecer los valores de los píxeles del componente de luminancia de la imagen VSP en los píxeles de la partición de MB actual. Por ejemplo, la unidad de compensación de movimiento 164 puede generar un bloque predictivo de luminancia para la partición de MB actual de tal manera que cada muestra de luminancia en el bloque predictivo de luminancia coincida con una muestra de luminancia en una localización correspondiente en la imagen VSP.

[0136] Además, a partir de las coordenadas (x', y') con el tamaño de N/2xM/2 en el caso del formato de vídeo 4:2:0, NxM en el caso del formato de vídeo 4:4:4 , o NxM/2 en el caso del formato de vídeo 4:2:2, la unidad de compensación de movimiento 164 puede establecer los valores de píxeles en cada bloque de un componente de crominancia de la imagen VSP en los píxeles de la partición de MB actual, comenzando en las mismas coordenadas de los componentes de crominancia (x', y'). Por ejemplo, la unidad de compensación de movimiento 164 puede generar un bloque Cb o Cr predictivo para la partición de MB actual de tal manera que cada muestra Cb o Cb coincida con una muestra Cb o Cr en una situación correspondiente en la imagen VSP. En este ejemplo, la unidad de compensación de movimiento 164 puede comenzar a generar el bloque Cb o Cr predictivo a partir de las coordenadas (x ', y') de un bloque Cb o Cr de la imagen VSP. Además, en este ejemplo, el bloque Cb o Cr predictivo puede ser de tamaño N/2xM/2, NxM o nxM/2 si los bloques de crominancia de la imagen VSP se muestrean de forma insuficiente de acuerdo con el formato de vídeo 4:2:0, el formato de vídeo 4:4:4 o formato de vídeo 4:2:2, respectivamente.

[0137] Cuando vsp_mb_flag o skip_from_vsp_flag es igual a 1, se deduce que cada bloque del MB tiene mb_part_vsp_flag y sub mb_vsp_flag igual a 1. Cuando no se prediga todo un MB a partir de una imagen VSP (por ejemplo, cuando vsp_mb_flag es igual a 0), las particiones de MB del MB pueden tener valores diferentes de mb_part_vsp_flag.

[0138] En una solución alternativa de ejemplo, está restringido que no todas las particiones de MB pueden tener mb_part_vsp_flag igual a 1 si vsp_mb_flag es igual a 0. En otras palabras, el codificador de vídeo 20 no señala que cada partición de MB de un MB se predice a partir de la imagen VSP si el codificador de vídeo 20 ha señalado que el MB completo no se predice a partir de la imagen VSP. La señalización de todas las particiones de MB del MB se predice a partir de la imagen VSP arrojaría los mismos resultados que la señalización de que todo el MB se predice a partir de la imagen VSP. Por lo tanto, si todas las particiones de MB del MB se predijeron a partir de la imagen VSP, puede ser un desperdicio de bits generar las particiones de MB.

[0139] Como se describió brevemente más arriba, la unidad de decodificación por entropía 150 puede realizar operaciones de decodificación por entropía para decodificar elementos de sintaxis por entropía. Además, la unidad de codificación por entropía 118 del codificador de vídeo 20 puede realizar operaciones de codificación por entropía para codificar por entropía elementos de sintaxis. Cuando un codificador por entropía (tal como la unidad de codificación por entropía 118 y la unidad de decodificación por entropía 150) realiza codificación por entropía o decodificación por entropía (es decir, codificación por entropía) en un elemento de sintaxis, el codificador por entropía puede determinar un modelo de codificación. El codificador por entropía puede determinar un modelo de contexto basándose en información contextual (es decir, un contexto). El modelo de contexto puede indicar probabilidades de bins con valores particulares.

[0140] De acuerdo con la primera implementación de ejemplo de las técnicas de esta divulgación, un codificador por entropía puede mantener un contexto para VSPFlag. Como se indicó anteriormente, VSPFlag puede indicar si se predice una partición de MB a partir de una imagen VSP. El codificador por entropía puede usar el contexto para la codificación por entropía de vsp mb_flag, mb_part_vsp_flag, sub mb_vsp_flag y skip_from_vsp_flag. La unidad de decodificación por entropía 150 puede actualizar el contexto para VSPFlag basándose en cada vsp mb_flag, mb_part_vsp_flag, sub_mb_vsp_flag y skip_from _vsp_flag.

[0141] Antes de codificar por entropía el primer elemento de sintaxis de un fragmento, el codificador por entropía ejecuta un proceso de inicialización de contexto para inicializar las variables de contexto usadas en los elementos de sintaxis de codificación por entropía del fragmento. De acuerdo con las técnicas de esta divulgación, la probabilidad inicial para el contexto de fragmentos VSPFlag de P puede ser diferente de la probabilidad inicial para el contexto de VSPFlag de los fragmentos B, aunque el modo de VSP se aplica tanto a los fragmentos P como a los fragmentos B.

[0142] Además, cuando un decodificador por entropía comience a decodificar por entropía un elemento de sintaxis, el decodificador por entropía puede recuperar un conjunto de todas las binarizaciones posibles del elemento de sintaxis y puede establecer un índice de bin (binIdx) en -1. El decodificador por entropía puede procesar secuencialmente bits del flujo de bits. Cuando el decodificador por entropía procese un bit, el decodificador por entropía puede incrementar binIdx y puede determinar un índice de contexto (ctxIdx) basándose al menos en parte en binIdx. El decodificador por entropía puede usar un modelo de contexto asociado con el ctxIdx para decodificar por entropía el bin actual. A continuación, el decodificador por entropía puede determinar si los intervalos que se han decodificado hasta el momento para el elemento de sintaxis coinciden con cualquiera de las posibles binarizaciones del elemento de sintaxis. Si no, el decodificador por entropía puede procesar otro bit de la manera descrita en este párrafo. Cuando los bins decodificados coincidan con una posible binarización del elemento de sintaxis, el decodificador por entropía puede convertir la binarización del elemento de sintaxis en el valor original del elemento de sintaxis.

[0143] Un codificador por entropía puede realizar un proceso similar. En este proceso, el codificador por entropía puede binarizar un elemento de sintaxis y puede establecer un índice de bin (binIdx) en -1. El codificador por entropía puede procesar entonces los contenedores del elemento de sintaxis binarizado secuencialmente. Cuando el codificador por entropía procesa un bin, el codificador por entropía puede determinar, basándose al menos en parte en el binIdx para el bin, un índice de contexto (ctxIdx) para el bin. El codificador por entropía puede usar un modelo de contexto asociado con el ctxIdx para que el bin codifique por entropía el bin. Cuando el codificador por entropía ha procesado cada bin del elemento de sintaxis binario, el codificador por entropía ha completado la codificación por entropía del elemento de sintaxis.

[0144] Como se mencionó anteriormente, cuando un codificador por entropía (por ejemplo, el decodificador por entropía o el codificador por entropía) procesa un bin de un elemento de sintaxis, el codificador por entropía determina, basándose al menos en parte en un índice de bin (binIdx) del bin, un índice de contexto (ctxIdx) para el bin. Para determinar el ctxIdx para el bin, el codificador por entropía puede usar el índice de bin y un desplazamiento de índice de contexto (ctxIdxOffset) para buscar, en una tabla predeterminada, un incremento de índice de contexto (ctxIdxInc) para el bin. El codificador por entropía puede determinar el ctxIdxOffset como parte de la binarización del elemento de sintaxis. El ctxIdx para el bin puede ser igual al ctxIdxOffset más el ctxIdxInc para el bin.

[0145] De acuerdo con la primera implementación de ejemplo de las técnicas de esta divulgación, cuando un codificador por entropía esté codificando por entropía el VSPFlag (que puede ser el elemento de sintaxis skip_from vsp_flag, el elemento de sintaxis vsp_mb_flag_syntax, o el elemento de sintaxis sub mb_vsp_flag dependiendo del elemento de sintaxis que se vaya a decodificar), el codificador por entropía puede determinar el ctxIdxInc para un bin asociado con el VSPFlag. En algunos ejemplos, el ctxIdxInc para el bin puede ser la suma de condTermFlagA y condTermFlagB. Es decir, el ctxIdxInc se puede derivar como ctxIdxInc = condTermFlagA + con-dTermFlagB. En otro ejemplo, el ctxIdxInc para el bin puede ser igual a condTermFlagA. Es decir, el ctxIdxInc se puede derivar como ctxIdxInc = condTermFlagA. En otro ejemplo, el ctxIdxInc para el bin es igual a condTermFlagB. Es decir, el ctxIdxInc se puede derivar como ctxIdxInc = condTermFlagB.

[0146] En los ejemplos anteriores, el codificador por entropía puede determinar condTermFlagA y condTermFlagB basándose al menos en parte en si los datos asociados con bloques particulares que están al lado de un MB o de una partición de MB actual actual están disponibles para el proceso de codificación por entropía para los elementos de sintaxis asociados con el MB o la partición de MB actual. Por ejemplo, mbPAddrA puede indicar el bloque vecino 8x8 a la izquierda del MB o de la partición de MB actual actual y mbPAddrB puede indicar el bloque vecino 8x8 encima del MB o de la partición de MB actual actual. Si mbPAddrA no está disponible o el VSPFlag para el bloque mbPAddrA es igual a 0, condTermFlagA es igual a 0. De lo contrario, si mbPAddrA está disponible y el VSPFlag para el bloque mbPAddrA es igual a 1, condTermFlagA es igual a 1. De forma similar, si mbPAddrB no está disponible o el VSPFlag para el bloque mbPAddrB es igual a 0, condTermFlagB es igual a 0. De lo contrario, si mbPAddrB está disponible y el VSPFlag para el bloque mbPAddrB es igual a 1, condTermFlagB es igual a 1.

[0147] En otro ejemplo, si mbPAddrA está disponible y el VSPFlag para el bloque mbPAddrA es igual a 0, condTermFlagA es igual a 1. De lo contrario, si mbPAddrA no está disponible o el VSPFlag para el bloque mbPAddrA es igual a 1, condTermFlagA es igual a 0. De forma similar, si mbPAddrB está disponible y el VSPFlag para el bloque mbPAddrB es igual a 0, condTermFlagB es igual a 1. De lo contrario, si mbPAddrB no está disponible o el VSPFlag para el bloque mbPAddrB es igual a 1, condTermFlagB es igual a 0.

[0148] Como se mencionó brevemente anteriormente, la unidad de filtro 160 puede realizar una operación de desbloqueo para reducir los artefactos de bloqueo asociados con los bloques de muestra de la unidad de vídeo actual. De acuerdo con la primera implementación de ejemplo de las técnicas de esta divulgación, la unidad de filtro 160 puede realizar operaciones de desbloqueo particulares relacionadas con el modo de VSP. En un primer proceso de filtro de desbloqueo de ejemplo, una partición de MB con VSPFlag igual a 1 se considera intercodificación, con un índice de referencia igual a -1, unidireccionalmente intercodificado de la lista de imágenes de referencia 0 y un vector de movimiento igual a cero durante el proceso de filtro de desbloqueo. La unidad de filtro 160 puede determinar un valor de resistencia de filtrado límite (bS) para un límite entre la partición de MB y otro bloque. La unidad de filtro 160 puede usar el bS para aplicar un filtro a la matriz de muestras de entrada en el límite. En este primer ejemplo de proceso de filtro de desbloqueo, la unidad de filtro 160 puede determinar bS basándose al menos en parte en los

modos de predicción (por ejemplo, inter o intra) de los bloques, en los índices de referencia de los bloques, si los bloques se someten a interpredicción unidireccional o bidireccionalmente, y en los vectores de movimiento de los bloques.

[0149] En segundo proceso de filtro de desbloqueo de ejemplo, la unidad de filtro 160 puede ejecutar un proceso de derivación para el contenido de luminancia dependiente de la fuerza de filtrado límite como parte de un proceso de filtro de desbloqueo. En este ejemplo, las entradas al proceso de derivación son los valores de muestra de entrada p_0 y q_0 de un único conjunto de muestras a través de un borde que se vaya a filtrar. Las entradas al proceso de derivación también incluyen un verticalEdgeFlag. El VerticalEdgeFlag puede indicar si el borde es horizontal o vertical. La salida del proceso de derivación es una variable de resistencia límite bS.

[0150] En el segundo proceso de filtro de desbloqueo de ejemplo, una variable MbaffFrameFlag puede ser igual a 1 cuando un elemento de sintaxis mb_adaptive_frame_field_flag en un SPS sea igual a 1 y un field_pic_flag en una cabecera de fragmento no sea igual a 1. Es decir, MbaffFrameFlag = mb_adaptive_frame_field_flag &&! Field_pic_flag. El elemento de sintaxis mb_adaptive_frame_field_flag especifica si es posible conmutar entre MB de campo y de trama dentro de una imagen. Un campo es un conjunto de filas alternativas de una trama. Mb_adaptive_frame_field igual a 0 no especifica ninguna conmutación entre los MB de trama y de campo dentro de una imagen. Mb_adaptive_frame_field igual a 1 especifica el uso posible de conmutar entre MB de trama y de campo dentro de tramas. El elemento de sintaxis field_pic_flag indica si el fragmento es un fragmento de un campo codificado o de una trama codificada. Un campo codificado es una representación codificada de un campo. Field_pic_flag igual a 1 puede especificar que el fragmento es un fragmento de un campo codificado. Field_pic_flag igual a 0 puede especificar que el fragmento es un fragmento de una trama codificada. Por tanto, MbaffFrameFlag igual a 1 indica el uso posible de conmutar entre MB de trama y de campo dentro de tramas y que el fragmento actual es un fragmento de una trama codificada.

[0151] Además, en el segundo proceso de filtro de desbloqueo de ejemplo, la unidad de filtro 160 puede derivar una variable mixMo-deEdgeFlag de la siguiente manera. Primero, si MbaffFrameFlag es igual a 1 y las muestras p_0 y q_0 están en diferentes pares de MB, uno de los cuales es un par de MB de campo y el otro es un par de MB de trama, la unidad de filtro 160 establece que mixedModeEdgeFlag es igual a 1. De lo contrario, si MbaffFrameFlag no es igual a 1 o las muestras p_0 y q_0 no están en diferentes pares de MB, la unidad de filtro 160 establece mixedMo-deEdgeFlag igual a 0.

[0152] A continuación, en el segundo proceso de filtro de desbloqueo de ejemplo, la unidad de filtro 160 deriva la variable bS para el borde del bloque que se esté filtrando. Si el borde del bloque es un borde de MB y cualquiera de las siguientes condiciones es verdadera, se emite un valor de bS igual a 4. De acuerdo con una primera condición, la unidad de filtro 160 establece el valor de bS en 4 si las muestras p_0 y q_0 están ambas en MB de trama y si una o ambas muestras p_0 y q_0 están en un MB codificado usando un modo de predicción de MB intran. De acuerdo con una segunda condición, la unidad de filtro 160 establece el valor de bS en 4 si las muestras p_0 y q_0 están ambas en MB de trama y si una o ambas muestras p_0 o q_0 están en un MB que está en un fragmento con slice_type igual a SP o SI. De acuerdo con una tercera condición, la unidad de filtro 160 establece el valor de bS en 4 si MbaffFrameFlag es igual a 1 o field_pic_flag es igual a 1, si verticalEdgeFlag es igual a 1 y si una o ambas muestras p_0 o q_0 están en un MB codificado usando un modo de predicción de MB intran. De acuerdo con una cuarta condición, la unidad de filtro 160 puede establecer el valor de bS en 4 si MbaffFrameFlag es igual a 1 o field_pic_flag es igual a 1, si verticalEdgeFlag es igual a 1 y si una o ambas muestras p_0 o q_0 están en un MB que está en un fragmento con tipo de corte igual a SP o SI.

[0153] De lo contrario, si cualquiera de las siguientes condiciones es verdadera, la unidad de filtro 160 establece el valor de bS en 3. De acuerdo con una primera condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 0 y una o ambas muestras p_0 o q_0 están en un MB codificado usando un modo de predicción de MB Intra. De acuerdo con una segunda condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 0 y una o ambas muestras p_0 o q_0 están en un MB que está en un fragmento con un tipo de fragmento igual a SP o SI. De acuerdo con una tercera condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 1, si verticalEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 están en un código MB usando un modo de predicción de MB Intra. De acuerdo con una cuarta condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 1, si verticalEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 están en un MB que está en un fragmento con tipo de fragmento igual a SP o SI.

[0154] De lo contrario, si cualquiera de las siguientes condiciones es verdadera, la unidad de filtro 160 establece el valor de bS en 2. De acuerdo con una primera condición, la unidad de filtro 160 establece el valor de bS en 2 si transform_size_8x8_flag es igual a 1 para el MB que contiene la muestra p_0 y si el bloque de transformada de luminancia 8x8 asociado con el bloque de luminancia 8x8 que contiene la muestra p_0 contiene niveles de coeficiente de transformada distintos de cero. De acuerdo con una segunda condición, la unidad de filtro 160 establece el valor de bS en 2 si transform_size_8x8_flag es igual a 0 para el MB que contiene la muestra p_0 y si el bloque de transformada de luminancia 4x4 asociado con el bloque luminancia 4x4 que contiene la muestra p_0 contiene niveles de coeficientes de transformada distintos de cero. De acuerdo con una tercera condición, la unidad de filtro 160 establece el valor de bS en 2 si transform_size_8x8_flag es igual a 1 para el MB que contiene la muestra q_0 y si el bloque de transformada de luminancia 8x8 asociado con el bloque de luminancia 8x8 que contiene la muestra q_0 contiene niveles de coeficiente

de transformada distintos de cero. De acuerdo con una cuarta condición, la unidad de filtro 160 establece el valor de bS en 2 si transform_size_8x8_flag es igual a 0 para el MB que contiene la muestra q_0 y si el bloque de transformada de luminancia 4x4 asociado con el bloque de luminancia 4x4 que contiene la muestra q_0 contiene niveles de coeficientes de transformada distintos de cero.

[0155] De lo contrario, si cualquiera de las siguientes condiciones es verdadera, la unidad de filtro 160 establece que el valor de bS es igual a 1. De acuerdo con una primera condición, la unidad de filtro 160 establece el valor de bS en 1 si mixedModeEdgeFlag es igual a 1. De acuerdo con una segunda condición y de acuerdo con las técnicas de esta divulgación, la unidad de filtro 160 establece el valor de bS en 1 si mixedModeEdgeFlag es igual a 0 y si el valor del VSPFlag para la partición de MB/subMB que contiene la muestra p_0 es diferente del VSPFlag para la partición de MB/subMB que contiene la muestra q_0. De acuerdo con una tercera condición, la unidad de filtro 160 establece el valor de bS en 1 si mixedModeEdgeFlag es igual a 0 y si la predicción de la partición de MB/subMB que contiene diferentes imágenes de referencia o un número diferente de vectores de movimiento de la muestra p_0 se usan que para la predicción de la partición de MB/subMB que contiene la muestra q_0. La determinación de si las imágenes de referencia usadas para las dos particiones de MB/subMB son iguales o diferentes se basa únicamente en las imágenes a las que se hace referencia, independientemente de si se forma una predicción usando un índice en la lista de imágenes de referencia 0 o un índice en la lista de imágenes de referencia 1, y también sin tener en cuenta si la posición del índice dentro de una lista de imágenes de referencia es diferente. De acuerdo con una cuarta condición, la unidad de filtro 160 establece el valor de bS en 1 si mixedModeEdgeFlag es igual a 0 y se usa un vector de movimiento para predecir la partición de MB/subMB que contiene la muestra p_0 y se usa un vector de movimiento para predecir la partición de MB/subMB que contiene la muestra q_0 y la diferencia absoluta entre el componente horizontal o vertical de los vectores de movimiento usados es mayor o igual que 4 en las unidades de muestras de tramas de cuartos de luminancia. De acuerdo con una quinta condición, la unidad de filtro 160 establece el valor de bS en 1 si mixedModeEdgeFlag es igual a 0 y dos vectores de movimiento y dos imágenes de referencia diferentes se usan para predecir la partición de MB/subMB que contiene la muestra p_0 y dos vectores de movimiento para las mismas dos imágenes de referencia se usan para predecir la partición de MB/subMB que contiene la muestra q_0 y la diferencia absoluta entre el componente horizontal o vertical de los dos vectores de movimiento usados en la predicción de las dos particiones de MB/subMB para la misma imagen de referencia son mayores o iguales a 4 en las unidades de muestras de tramas de cuarto de luminancia.

[0156] De acuerdo con una sexta condición, la unidad de filtro 160 establece el valor de bS en 1 si mixedModeEdgeFlag es igual a 0 y dos vectores de movimiento para la misma imagen de referencia se usan para predecir la partición de MB/subMB que contiene la muestra p_0 y dos vectores de movimiento para la misma imagen de referencia se usan para predecir la partición de MB/subMB que contiene la muestra q_0 y las dos condiciones siguientes son verdaderas. En primer lugar, la diferencia absoluta entre el componente horizontal o vertical de los vectores de movimiento RefPicListO usados en la predicción de las dos particiones de MB/subMB es mayor o igual a 4 en las muestras de tramas de cuarto de luminancia o la diferencia absoluta entre el componente horizontal o vertical de los vectores de movimiento RefPicList1 usados en la predicción de las dos particiones de MB/subMB es mayor o igual a 4 en las unidades de muestras de trama de cuarto de luminancia. Segundo, la diferencia absoluta entre el componente horizontal o vertical del vector de movimiento RefPicListO usado en la predicción de la partición de MB/subMB que contiene la muestra p_0 y el vector de movimiento RefPicList1 usado en la predicción de la partición de MB/subMB que contiene la muestra q_0 es mayor o igual que 4 en las unidades de muestras de cuarto de luminancia o la diferencia absoluta entre el componente horizontal o vertical del vector de movimiento RefPicList1 usado en la predicción de la partición de MB/subMB que contiene la muestra p_0 y el vector de movimiento RefPicListO usado en la predicción de la partición de MB/subMB que contiene la muestra q_0 es mayor o igual que 4 en las unidades de las muestras de trama de cuarto de luminancia. Una diferencia vertical de 4 en las unidades de muestras de tramas de cuarto de luminancia es una diferencia de 2 en las unidades de muestras de tramas de cuarto de luminancia. De lo contrario, la unidad de filtro 160 establece el valor de bS igual a 0.

[0157] En un tercer ejemplo de proceso de filtro de desbloqueo, la unidad de filtro 160 puede realizar un proceso de derivación para la intensidad del filtrado de límite dependiente del contenido de luminancia como parte de la ejecución de una operación de desbloqueo. En este ejemplo, las entradas al proceso de derivación son los valores de muestra de entrada p_0 y q_0 de un único conjunto de muestra a través de un borde que se deba filtrar. Las entradas al proceso de derivación también incluyen un verticalEdgeFlag. La salida del proceso de derivación puede ser una variable de resistencia de límite bS. En este ejemplo, la unidad de filtro 160 puede derivar una variable mixedModeEdgeFlag de la siguiente manera. Primero, si MbaffFrameFlag es igual a 1 y las muestras p_0 y q_0 están en pares diferentes, uno de los cuales es un par de MB de campo y el otro es un par de MB de trama, la unidad de filtro 160 establece que mixedModeEdgeFlag es igual a 1. De lo contrario, si MbaffFrameFlag no es igual a 1 o las muestras p_0 y q_0 no están en diferentes pares de MB, la unidad de filtro 160 establece que mixedModeEdgeFlag es igual a 0.

[0158] Además, en el tercer proceso de filtro de desbloqueo de ejemplo, la unidad de filtro 160 puede derivar una variable bS para un borde de bloque entre MB. Si el borde del bloque es un borde de MB y cualquiera de las siguientes condiciones es verdadera, se emite un valor de bS igual a 4. De acuerdo con una primera condición, la unidad de filtro 160 establece el valor de bS en 4 si las muestras p_0 y q_0 están ambas en MB de tramas y una o ambas muestras p_0 y q_0 está en un MB codificado usando un modo de predicción de MB intra. De acuerdo con un segundo estado y de acuerdo con las técnicas de esta divulgación, la unidad de filtro 160 establece el valor de bS a 4 si las muestras p_0 y

q_0 están ambas en MB de trama y una o ambas de p_0 o q_0 de muestra están en un MB codificado con VSPFlag igual a 1. De acuerdo con una tercera condición, la unidad de filtro 160 establece el valor de bS en 4 si las muestras p_0 y q_0 están ambas en MB de trama y una o ambas muestras p_0 o q_0 están en un MB que está en un fragmento con slice_type igual a SP o SI. De acuerdo con una cuarta condición, la unidad de filtro 160 establece el valor de bS en 4 si MbaffFrameFlag es igual a 1 o si field_pic_flag es igual a 1, si verticalEdgeFlag es igual a 1 y si una o ambas muestras p_0 o q_0 están en un MB codificado usando un modo de predicción de MB intra. De acuerdo con una quinta condición y de acuerdo con las técnicas de esta divulgación, la unidad de filtro 160 establece el valor de bS en 4 si MbaffFrameFlag es igual a 1 o si field_pic_flag es igual a 1 y si verticalEdgeFlag es igual a 1, y si una o ambas de las muestras p_0 o q_0 están en un MB codificado con VSPFlag igual a 1. De acuerdo con una sexta condición, la unidad de filtro 160 puede establecer el valor de bS en 4 si MbaffFrameFlag es igual a 1 o si field_pic_flag es igual a 1, si verticalEdgeFlag es igual a 1 y si una o ambas muestras p_0 o q_0 es en un MB que está en un fragmento con slice_type igual a SP o SI.

[0159] De lo contrario, en el tercer proceso de filtro de desbloqueo de ejemplo, si cualquiera de las siguientes condiciones es verdadera, la unidad de filtro 160 establece el valor de bS en 3. De acuerdo con una primera condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 están en un MB codificado usando un modo de predicción de MB Intra. De acuerdo con una segunda condición y de acuerdo con las técnicas de esta divulgación, la unidad de filtro 160 establece el valor de bS en 3 si el ModeEdgeFlag mixto es igual a 0 y si una o ambas muestras p_0 o q_0 están en un MB codificado con VSPFlag igual a 1. De acuerdo con una tercera condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 están en un MB que esté en un fragmento con slice_type igual a SP o SI. De acuerdo con una cuarta condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 1, si verticalEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 están en un código de MB usando un modo de predicción de MB Intra. De acuerdo con una quinta condición y de acuerdo con las técnicas de esta divulgación, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 0, si verticalEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 están en un MB codificado con VSPFlag igual a 1. De acuerdo con una sexta condición, la unidad de filtro 160 establece el valor de bS en 3 si mixedModeEdgeFlag es igual a 1, si verticalEdgeFlag es igual a 0 y si una o ambas muestras p_0 o q_0 está en un MB que esté en un fragmento con slice_type igual a SP o SI.

[0160] De lo contrario, en el tercer proceso de filtro de desbloqueo de ejemplo, la unidad de filtro 160 puede establecer el valor de bS a 2 si cualquiera de las condiciones descritas anteriormente en el segundo proceso de filtro de desbloqueo de ejemplo están satisfechas. De lo contrario, la unidad de filtro 160 puede establecer el valor de bS en 1 si se cumple alguna de las condiciones descritas anteriormente en el segundo proceso de filtro de desbloqueo de ejemplo, excepto la segunda condición. De lo contrario, la unidad de filtro 160 puede establecer el valor de bS en 0.

[0161] Una segunda implementación de ejemplo de las técnicas de esta divulgación usa diferentes sintaxis y semánticas que el primer ejemplo descrito anteriormente. En el segundo ejemplo de implementación, la sintaxis y la semántica se cambian de tal manera que mb_skip_flag y skip_from_vsp_flag se señalan de manera condicional y flexible. En esta segunda implementación de ejemplo, la sintaxis para la capa de MB, la predicción de MB y la predicción de subMB pueden ser las mismas que en la primera implementación de ejemplo descrita anteriormente. Sin embargo, en la segunda implementación de ejemplo, la sintaxis y la semántica de los datos de fragmento y la semántica correspondiente pueden ser diferentes que en la primera implementación de ejemplo.

[0162] La Tabla 6 siguiente es una sintaxis de ejemplo para una estructura de sintaxis de datos de fragmento de acuerdo con la segunda implementación de ejemplo.

TABLA 6 - Sintaxis de datos de fragmento

slice_data() {	C	Descriptor
if(entropy_coding_mode_flag)		
while(!byte_aligned())		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		

slice_data() {	C	Descriptor
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data()		
} else {		
if(slice_vsp_flag && MbSkipFromVSPFlagLeft && MbSkipFromVSPFlagUp) {		
if(VspRefExist){		
skip_from_vsp_flag	2	ae(v)
moreDataFlag = ! skip_from_vsp_flag		
}		
if(!skip_from_vsp_flag) {		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
} else {		
mb_skip_flag	2	ae(v)
if (slice_vsp_flag && !mb_skip_flag && VspRefExist)		
skip_from_vsp_flag	2	ae(v)
moreDataFlag = !skip_from_vsp_flag && !mb_skip_flag		
}		
}		
if(moreDataFlag) {		
if(MbaffFrameFlag && (CurrMbAddr % 2 = = 0 || (CurrMbAddr % 2 = = 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) | ae(v)
macroblock_layer()	2 | 3 | 4	
}		
if(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data()		
else {		
if(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
if(MbaffFrameFlag && CurrMbAddr % 2 = = 0)		
moreDataFlag = 1		
else {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		

slice_data() {	C	Descriptor
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} while(moreDataFlag)		
}		

[0163] En el ejemplo de sintaxis de la Tabla 6, y de acuerdo con las técnicas de esta divulgación, una estructura de sintaxis de datos de fragmento para un fragmento puede incluir un elemento de sintaxis skip_from_vsp_flag para un MB actual cuando la VSP esté habilitada para el fragmento, una variable MbSkipFromVSPFlagLeft = 1 y una variable MbSkipFromVSPFlagUp = 1, y RefPicListO o RefPicList1 incluye una imagen VSP. En otros ejemplos, la estructura de sintaxis de datos de fragmento para un fragmento puede incluir un elemento de sintaxis skip_from_vsp_flag para un MB actual cuando una variable MbSkipFromVSPFlagLeft = 1 y una variable MbSkipFromVSPFlagUp = 1, y RefPicListO o RefPicList1 incluye una imagen VSP. Por ejemplo, la estructura de sintaxis de datos de fragmento puede incluir skip_from_vsp_flag cuando slice_vsp_flag = 1, MbSkipFromVSPFlagLeft = 1, MbSkipFromVSPFlagUp = 1 y VspRefExist = 1. En algunos ejemplos, MbSkipFromVSPFlagLeft y MbSkipFromVSPFlagUp se pueden establecer en los valores de skip_from_vsp_flag de los MB disponibles superior y izquierdo, respectivamente. En un ejemplo alternativo, MbSkipFromVSPFlagLeft se establece en 1 si cualquiera de los bloques disponibles en el MB izquierdo tiene VSPFlag igual a 1. En este ejemplo, MbSkipFromVSPFlagUp se establece en 1, si cualquiera de los bloques disponibles en el MB anterior tiene VSPFlag igual a 1. En otro ejemplo alternativo, MbSkipFromVSPFlagLeft se establece en 1 si ambos bloques 8x8 quedan disponibles y tienen VSPFlag igual a 1. En este ejemplo, MbSkipFromVSPFlagUp se establece en 1 si ambos bloques superiores a 8x8 están disponibles y tienen VSPFlag igual a 1.

[0164] skip_from_vsp_flag para un MB actual indica si el MB actual completo se predice a partir de una imagen VSP, el MB actual está asociado con bloques de coeficientes de transformada de luminancia y/o de crominancia que incluyen niveles de coeficientes de transformada distintos de cero y el elemento de sintaxis mb_type para el MB actual está presente. Por ejemplo, skip_from_vsp_flag igual a 1 indica que todo el MB actual se predice a partir de una imagen VSP y CodedBlockPatternLuma y CodedBlock-PatternChroma se establecen en 0, y mb_type no está presente. Cuando skip_from_vsp_flag es igual a 0, el MB actual no se predice a partir de una imagen VSP o al menos uno de CodedBlockPatternLuma y CodedBlockPatternChroma no es igual a 0. De forma alternativa, cuando skip_from_vsp_flag es igual a 0, el MB actual se predice usando el modo de omisión normal. Cuando skip_from_vsp_flag no está presente, se puede deducir que skip_from_vsp_flag es igual a 0. En algunos ejemplos alternativos, cuando skip_from_vsp_flag es igual a 1, puede deducirse que mb_skip_flag, independientemente de si mb_skip_flag está presente o no, es igual a 1.

[0165] En la sintaxis de ejemplo de la Tabla 6, mb_skip_flag igual a 1 especifica que para el MB actual, cuando se decodifica un fragmento P o SP, se deduce que mb_type es P_Skip y el tipo de MB se denomina colectivamente tipo P de MB, o para el cual, al decodificar un fragmento B, se debe deducir que mb_type es B_Skip y el tipo MB se denomina colectivamente tipo B de MB. mb_skip_flag igual a 0 especifica que no se omite el MB actual.

[0166] Además, en la sintaxis de ejemplo de la Tabla 6, una variable moreDataFlag es igual a 1 si skip_from_vsp_flag es igual a 0 y mb_skip_flag es igual a 0 (es decir, moreDataFlag =! Skip_from_vsp_flag &&! Mb_skip_flag). Si moreDataFlag es igual a 1, la estructura de sintaxis de los datos del fragmento puede incluir una estructura de sintaxis de capa de macrobloque para el MB actual. En otros ejemplos, moreDataFlag puede ser igual a 1 si mb_skip_flag es igual a 0 (es decir, moreDataFlag =! Mb_skip_flag).

[0167] En la segunda implementación de ejemplo, un codificador por entropía, tal como la unidad de codificación por entropía 118 o la unidad de decodificación por entropía 150, puede ejecutar un proceso de codificación por entropía similar al proceso de codificación por entropía descrito con respecto a la primera implementación de ejemplo. De forma alternativa, en la segunda implementación de ejemplo, el codificador por entropía puede ejecutar una inicialización de contexto diferente y diferentes operaciones de selección de contexto que las descritas anteriormente con respecto a la primera implementación de ejemplo.

[0168] Por ejemplo, en la segunda implementación de ejemplo, el codificador por entropía puede usar dos conjuntos de contextos, uno para skip_from_vsp_flag y el otro para vsp_mb_flag, mb_part_vsp_flag y sub_mb_vsp_flag. Durante la operación de inicio de contexto, el codificador por entropía puede inicializar las probabilidades iniciales de los contextos de vsp mb_flag, mb_part_vsp_flag o sub mb_vsp_flag con diferentes valores para los fragmentos P y para los fragmentos B. El modo de VSP puede aplicarse tanto para los fragmentos P como para los fragmentos B. Además, el codificador por entropía puede inicializar las probabilidades iniciales para el contexto de skip_from_vsp_flag de los fragmentos P de forma diferente de los fragmentos B.

[0169] En la segunda implementación de ejemplo, cuando el codificador por entropía esté codificando por entropía un mb_part_sp_flag, un sub_mb_vsp_flag, un vsp_mb_flag o un skip_from_vsp_flag, el codificador por entropía puede determinar un ctxIdxInc para un bin asociado con uno de estos elementos de sintaxis. El codificador por entropía puede determinar el ctxIdxInc para el bin basándose al menos en parte en condTermFlagA y condTermFlagB. En diversos ejemplos, el codificador por entropía puede determinar el ctxIdxInc para el bin de tal manera que ctxIdxInc = condTermFlagA + condTermFlagB, de tal manera que ctxIdxInc = condTermFlagA, o de tal manera que ctxIdxInc = condTermFlagB. En la segunda implementación de ejemplo, el codificador por entropía puede determinar condTermFlagA y condTermFlagB basándose al menos en parte en si los datos asociados con bloques particulares que están al lado de una partición de MB o de un MB están disponibles para el proceso de codificación por entropía para los elementos de sintaxis asociados con el MB o con la la partición de MB actual.

[0170] Por ejemplo, en la segunda implementación de ejemplo, si el codificador por entropía está codificando un mb_part_vsp_flag, un sub_mb_vsp_flag, un elemento de sintaxis vsp_mb_flag, mbPAddrA puede indicar el bloque vecino 8x8 a la izquierda del MB o de la partición de MB actual y mbPAddrB puede indicar el bloque vecino 8x8 encima del MB o de la partición de MB actual. En este ejemplo, si mbPAddrA no está disponible o NonSkipVSPFlag para mbPAddrA es igual a 0, condTermFlagA es igual a 0. En este ejemplo, NonSkipVSPFlag puede ser el vsp_mb_flag, el mb_part_vsp_flag o el sub mb_vsp_flag dependiendo del elemento de sintaxis que se vaya a codificar. De lo contrario, si mbPAddrA está disponible y NonSkipVSPFlag para mbPAddrA es igual a 1, condTermFlagA es igual a 1. De forma similar, si mbPAddrB no está disponible o NonSkipVSPFlag para mbPAddrB es igual a 0, condTermFlagB es igual a 0. Si mbPAddrB está disponible y NonSkipVSPFlag para mbPAddrB es igual a 1, condTermFlagB es igual a 1.

[0171] Además, en la segunda implementación de ejemplo, si el codificador por entropía está codificando un elemento de sintaxis skip_from_vsp_flag, mbPAddrA puede indicar el bloque vecino 8x8 a la izquierda del MB o de la partición de MB actual actual y mbPAddrB puede indicar el bloque vecino 8x8 encima del MB o de la partición de MB actual. En este ejemplo, si mbPAddrA está disponible y skip_from_vsp_flag para mbPAddrA es igual a 0, condTermFlagA puede ser igual a 1. De lo contrario, si mbPAddrA no está disponible o skip_from_vsp_flag para mbPAddrA es igual a 1, condTermFlagA puede ser igual a 0. De forma similar, si mbPAddrB está disponible y skip_from_vsp_flag para mbPAddrB es igual a 0, condTermFlagB puede ser igual a 1. De lo contrario, si mbPAddrB no está disponible o skip_from_vsp_flag para mbPAddrB es igual a 1, condTermFlagB puede ser igual a 0.

[0172] En otro ejemplo donde el codificador por entropía esté codificando skip_from_vsp_flag, si mbPAddrA está disponible y skip_from_vsp_flag para mbPAddrA es igual a 1, condTermFlagA puede ser igual a 1. De lo contrario, si mbPAddrA no está disponible o skip_from_vsp_flag para mbPAddrA es igual a 0, condTermFlagA puede ser igual a 0. De forma similar, si mbPAddrB está disponible y skip_from_vsp_flag para mbPAddrB es igual a 1, condTermFlagB puede ser igual a 1. De lo contrario, si mbPAddrB no está disponible o skip_from_vsp_flag para mbPAddrB es igual a 0, condTermFlagB puede ser igual a 0.

[0173] En otro ejemplo en el que el codificador por entropía esté codificando skip_from_vsp_flag, si mbPAddrA está disponible y vsp mb_flag para mbPAddrA es igual a 0, condTermFlagA puede ser igual a 1. De lo contrario, si mbPAddrA no está disponible o vsp mb_flag para mbPAddrA es igual a 1, condTermFlagA puede ser igual a 0. De forma similar, si mbPAddrB está disponible y el vsp_mb_flag para mbPAddrB es igual a 0, condTermFlagB puede ser igual a 1. De lo contrario, si mbPAddrB no está disponible o vsp mb_flag para mbPAddrB es igual a 1, condTermFlagB puede ser igual a 0.

[0174] En otros ejemplos, el codificador por entropía puede determinar condTermFlagA y condTermFlagB tanto de skip_from_vsp_flag como de vsp_mb_flag. En otro ejemplo, el codificador por entropía puede determinar condTermFlagA y condTermFlagB a partir de vsp_mb_flag.

[0175] En una tercera implementación de ejemplo de las técnicas de esta divulgación, la sintaxis para la capa de MB, la predicción de subMB y los datos de fragmento pueden ser los mismos que en la primera implementación de ejemplo descrita anteriormente. Sin embargo, en la tercera implementación de ejemplo, la sintaxis puede optimizarse aún más basándose en diversas situaciones. Por ejemplo, la sintaxis y la semántica de la estructura de sintaxis de predicción de MB y la semántica correspondiente pueden ser diferentes que en la primera implementación de ejemplo. La Tabla 7 siguiente es una sintaxis de ejemplo para los datos de fragmento de acuerdo con la tercera implementación de ejemplo.

TABLA 7 - Sintaxis de predicción de macrobloque

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_ type, 0) = = Intra 16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		

mb_pred(mb_type) {	C	Descriptor
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if (slice_vsp_flag && NumMbPart(mb_type) != 1 && VspRefExist) //vsp_ mb_flag is not 1		
mb_part_vsp_flag[mbPartIdx **]**	2	ae(v)
if(!mb_part_vsp_flag[mbPartIdx] &&(num_ref_idx_10_active_minus1>0 || mb_field_decoding_flag) && MbPartPredMode(mb type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx]	2	te(v) | ae(v)
for(!mb_part_vsp_flag[mbPartIdx] &&(mbPartIdx = 0; mbPartIdx < NumMbPart(mb type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_ decoding_flag) && MbPartPredMode(mb_ _type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag[mbPartIdx] &&(MbPartPredMode (mb type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][**0**][**compIdx**]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag[mbPartIdx] &&(MbPartPredMode(mb_ type, mbPartIdx) != Pred L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

[0176] La Tabla 7 es similar a la Tabla 4 anterior. Sin embargo, la sintaxis de ejemplo de la Tabla 7 indica "para (! Mb_part_vsp_flag [mbPartIdx] && (mbPartIdx = 0; mbPartIdx <NumMbPart (mb_type); mbPartIdx ++)" en lugar de "para (mbPartIdx = 0; mbPartIdx <NumMbPart (mb_type) &&! mb_part_vsp_flag [mbPartIdx]; mbPartIdx ++)" en el bucle para generar/decodificar elementos de sintaxis ref_idx_l1. En otros ejemplos, la estructura de sintaxis de predicción de MB puede incluir mb_part_vsp_flag [mbPartIdx] si slice_vsp_flag && NumMbPart (mb_type)! = 1 && MbPartPredMode (mb_type, mbPartIdx) == Pred_L0.

[0177] Una cuarta implementación de ejemplo de las técnicas de esta divulgación puede ser similar a la segunda implementación de ejemplo descrita anteriormente. Sin embargo, en la cuarta implementación de ejemplo, el diseño de sintaxis de capa de MB, la predicción de MB y la predicción de subMB pueden cambiarse en relación con la segunda implementación de ejemplo. La Tabla 8 siguiente es una sintaxis de ejemplo de la estructura de sintaxis de capa de MB de acuerdo con la cuarta implementación de ejemplo.

TABLA 8 - Sintaxis de capa de macrobloque

macroblock_layer() {	C	Descriptor
if(slice_vsp_flag && MbVSPFlagLeft && bMbVSPFlagUp){		
if(VspRefExist)		
vsp_mb_flag	2	ae(v)
if (! vsp_mb_flag)		
mb_type	2	ue(v) | ae(v)
}		
else {		
mb_type	2	ue(v) | ae(v)
if (slice_vsp_flag && VspRefExist && (mb_type = = B_L0_16x16 || mb_type == P_L0_16x16))		
vsp_mb_flag	2	ae(v)
}		
if(mb_type == I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	3	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i]	3	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i **]**	3	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		

if(mb_type != I_NxN &&
MbPartPredMode(mb_type, 0) != Intra_16x16 &&
NumMbPart(mb_type) == 4) {		

if (!vsp_mb_flag)		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type = = I_NxN)		
transform_size_8x8_flag	2	u(1) | ae(v)
if (!vsp_mb_flag)		
mb_pred(mb_type)	2	
}		

macroblock_layer() {	C	Descriptor
if(MbPartPredMode(mb_type, 0) != Intra_16x16) {		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (vsp_mb_flag || (!vsp_mb_flag && (mb_type != B_Direct_16x16 || direct_ 8x8_ inference_flag))))		
transform_size_8x8_flag	2	u(1) | ae(v)
}		
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual(0, 15)	3 | 4	
}		
}		
}		

[0178] En el ejemplo de la Tabla 8, el contenido de una estructura de sintaxis de capa de MB depende de las variables MbVSPFlagLeft y MbVSPFlagUp. MbVSPFlagLeft y MbVSPFlagUp se pueden establecer en el valor de vsp mb_flag de los MB que están a la izquierda y encima del MB actual, respectivamente. De forma alternativa, MbVSPFlagLeft y MbVSPFlagUp se pueden establecer en el mayor de skip_from_vsp_flag y vsp mb_flag para los MB que están a la izquierda y encima del MB actual, respectivamente. Es decir, MbVSPFlagLeft se puede establecer en max (skip_from_vsp_flag, vsp_mb_flag) del MB izquierdo y MbVSPFlagUp se puede establecer en max (skip_from_vsp_flag, vsp_mb_flag) del MB superior. En otro ejemplo alternativo, MbVSPFlagLeft se establece en 1 si cualquiera de los bloques disponibles a la izquierda del MB actual tiene VSPFlag igual a 1. En este ejemplo, MbVSPFlagUp puede establecerse en 1 si cualquiera de los bloques disponibles encima del MB actual tiene VSPFlag igual a 0. En otro ejemplo más, MbVSPFlagLeft se establece en 1 si ambos bloques 8x8 a la izquierda del MB actual están disponibles para su uso en la codificación del MB actual y tienen VSPFlags igual a 1. En este ejemplo, MbVSPFlagUp se establece en 1 si ambos bloques 8x8 encima del MB actual están disponibles para su uso en la codificación del MB actual y tienen VSPFlags igual a 1.

[0179] En el ejemplo de la Tabla 8, la estructura de sintaxis de capa de MB puede incluir vsp mb_flag si slice_vsp_flag es igual a 1, si MbVSPFlagLeft es igual a 1 y si bMbVSPFlagUp es igual a 1 y si VspRefExist es igual a 1. Si slice_vsp_flag es igual a 1, si MbVSPFlagLeft es igual a 1 y si bMbVSPFlagUp es igual a 1 y si VspRefExist no es igual a 1, la estructura de sintaxis de capa de MB puede incluir mb_type. En otros ejemplos, la estructura de sintaxis de capa de MB puede incluir vsb mb_flag si MbVSPFlagLeft es igual a 1, y si bMbVSPFlagUp es igual a 1 y si VspRefExist es igual a 1. Si MbVSPFlagLeft es igual a 1, si bMbVSPFlagUp es igual a 1 y si VspRefExist no es igual a 1, la estructura de sintaxis de capa de MB puede incluir mb_type.

[0180] Además, en el ejemplo de la Tabla 8, la estructura de sintaxis de capa de MB puede incluir vsp_mb_flag si slice_vsp_flag && VspRefExist && (mb_type = = B_L0_16x16 || mb_type == P_L0_16x16). En otros ejemplos, la estructura de sintaxis de capa de MB puede incluir vsp_mb_flag si mb_type = = (B_L0_16x16 || mb_type == P_L0_16x16).

[0181] Además, la cuarta implementación de ejemplo, la sintaxis de la estructura de sintaxis de predicción de MB puede ser la misma que la mostrada en la Tabla 7 anterior. Sin embargo, en algunos ejemplos, la estructura de sintaxis de predicción de MB puede incluir mb_part_vsp_flag [mbPartIdx] si (slice_vsp_flag && NumMbPart (mb_type)! = 1 y MbPartPredMode (mb_type, mbPar-tIdx) == Pred_L0. La Tabla 9 siguiente muestra una sintaxis de predicción de subMB de ejemplo usada en la cuarta implementación de ejemplo.

TABLA 9 - Sintaxis de predicción de submacrobloque

sub mbpred(mb_ type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++) {		

sub mbpred(mb_ type) {	C	Descriptor
if (slice_vsp_flag) {// vsp mb flag is not 1		
if(SubMbVSPFlagLeft && SubMbVSPFlagUp){		
if(V spRefExist)		
sub_mb_vsp_flag[mbPartIdx **]**	2	ae(v)
if(!sub_ mb_vsp_flag**[** mbPartIdx **]**)		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
} else {		
sub_mb_type[mbPartIdx **]**	2	ae(v)
if (slice_vsp_flag && VspRefExist && (NumSubMbPart (sub_mb_type[mbPartIdx]) = = 1 || sub_mb_type[mbPartIdx] = = B_Direct_8x8))		
sub_mb_vsp_flag[mbPartIdx **]**	2	ae(v)
}		
}		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[** mbPartIdx **]**) &&(num_ref_idx_10_active_minus1 > 0 || mb_ field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type [mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[** mbPartIdx **]**) && (num_ref_idx_l1_active_minus1 > 0 || mb_ field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_ mb_ type[mbPartIdx **]**) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[** mbPartIdx **]**) &&(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_ mb_ type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_ mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[** mbPartIdx **]**) &&sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_ mb_ type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

[0182] En el ejemplo de la Tabla 9, las variables SubMbVSPFlagLeft y SubMbVSPFlagUp controlan el contenido de la predicción de subMB. SubMbVSPFlagLeft y SubMbVSPFlagUp se establecen en el valor de sub_mb_vsp_flag de los bloques izquierdo 8x8 y superior 8x8, respectivamente. El bloque izquierdo 8x8 está a la izquierda de la partición de subMB actual. El bloque 8x8 superior está por encima de la partición de subMB actual. En particular, en la sintaxis de ejemplo de la Tabla 9, la estructura de sintaxis de predicción de subMB puede incluir sub_mb_vsp_flag para una partición de subMB cuando SubMBVspFlagLeft y SubMBVspFlagUp sean iguales a 1 y RefPicListO o RefPicList1 incluya una imagen VSP (por ejemplo, VspRefExist = 1). sub mb_vsp_flag para la partición de subMB indica si las particiones de subMB se predicen desde una VSP. Además, en la sintaxis de ejemplo de la Tabla 9, la estructura de sintaxis de predicción de subMB puede incluir sub_mb_type para una partición de subMB cuando sub mb_vsp_flag para la partición de subMB indique que la partición de subMB no se predice a partir de una VSP.

[0183] Además, en la sintaxis de ejemplo de la Tabla 9, si SubMbVSPFlagLeft o SubMbVSPFlagUp no es igual a 1, la estructura de sintaxis de predicción de subMB puede incluir un sub_mb_vsp_flag para una partición de subMB cuando la VSP esté habilitada para un fragmento, RefPicListO o RefPicList1 incluya una imagen VSP y el número de particiones de subMB de la partición de subMB sea 1 o el tipo de subMB de la partición de subMB sea B_direct_8x8. Si el tipo de subMB de la partición de subMB es B_direct_8x8, la partición de subMB se predice bidireccionalmente y está particionada en bloques de 8x8.

[0184] En otro ejemplo, subMbVSPFlagLeft y SubMbVSPFlagUp se pueden establecer en el valor del VSPFlag de los bloques izquierdo 8x8 y superior 8x8, si están disponibles para su uso en la codificación de la partición de subMB actual, respectivamente.

[0185] Además, en una versión alternativa de la Tabla 9, la estructura de sintaxis de predicción de subMB puede incluir sub_mb_vsp_flag[mbPartIdx] si sub_mb_type[mbPartIdx] == P_L0_8x8 o sub_mb_type[mbPartIdx] == B_L0_8x8.

[0186] Una quinta implementación de ejemplo de las técnicas de esta divulgación es similar a la primera implementación de ejemplo descrita anteriormente. En esta quinta implementación de ejemplo, mb_skip_flag y skip_from_vsp_flag pueden combinarse en un elemento de sintaxis. Como se indicó anteriormente, mb_skip_flag indica si se usa el modo de omisión. skip_from_vsp_flag indica que, cuando se usa el modo de omisión, todo el MB se predice a partir de una imagen VSP. Por lo tanto, en la implementación del quinto ejemplo, el indicador combinado indica si se usa el modo de omisión y, si se usa el modo de omisión, que todo el MB se predice a partir de una imagen VSP.

[0187] Un sexto ejemplo de implementación de las técnicas de esta divulgación es similar al segundo ejemplo de implementación descrito anteriormente. Sin embargo, en la sexta implementación de ejemplo, las estructuras sintácticas de capa de MB y las estructuras de sintaxis de predicción de MB se pueden definir de acuerdo con las sintaxis de las Tablas 10 y 11, a continuación.

TABLA 10 - Estructura sintáctica de capa de macrobloque

macroblock_layer() {	C	Descriptor
mb_type	2	ue(v) | ae(v)
if(mb_type == B_Direct_16x16 && slice_vsp_flag && VspRefExist)		
vsp_direct_flag		
if(mb_type = = I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	3	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i**]**	3	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i **]**	3	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		
if(mb_type != I_NxN &&		

macroblock_layer() {	C	Descriptor
MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) = = 4) {		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type = = I_NxN)		
transform_size_8x8 flag	2	u(1) | ae(v)
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_ 16x16) {		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (**vsp_direct_flag**||(!**vsp_direct_flag** && (mb_type != B_Direct_16x16 || direct_8x8_inference_flag))))		
transform_size_8x8_flag	2	u(1) | ae(v)
}		
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_ type, 0) = = Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual(0, 15)	3 | 4	
}		
}		
}		

TABLA 11 - Sintaxis de predicción de macrobloque

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_ type, 0) = = Intra 16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		

mb_pred(mb_type) {	C	Descriptor
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if (slice_vsp_flag && VspRefExist)		
mb_part_vsp_flag[mbPartIdx **]**	2	ae(v)
if(!**mb_part_vsp_flag[** mbPartIdx **]** &&(num_ref_idx_10_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(!**mb_part_vsp_flag[** mbPartIdx **]** &&(mbPartIdx = 0; mbPartIdx < NumMbPart (mb_type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!**mb_part_vsp_flag[** mbPartIdx] &&(MbPartPredMode (mb type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][**0**][**compIdx]	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!**mb_part_vsp_flag[** mbPartIdx **]** &&(MbPartPredMode(mb_ type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

[0188] En la sintaxis de ejemplo de la Tabla 10, la estructura de sintaxis de capa de MB para un MB de un fragmento puede incluir un elemento de sintaxis de indicador vsp_direct si el tipo de MB es B_Direct_16x16 y VSP está habilitado para el fragmento y RefPicListO o RefPicList1 incluye una imagen VSP. Si el tipo de MB es B_Direct_16x16, el MB está en un fragmento B, se codifica en modo directo (es decir, no se señala ninguna información de movimiento, pero se señala la información residual) y el MB no está particionado en particiones de MB más pequeñas. El vsp_direct_flag puede indicar si el MB se predice a partir de una imagen VSP. Por ejemplo, vsp_direct_flag igual a 1 puede indicar que el MB con el modo B_direct_16x16 se predice a partir de la trama VSP y el indicador vsp_direct igual a 0 puede indicar que se usa el modo B_direct_16x16 original. Además, como se muestra en la Tabla 10, el vsp_direct_flag puede controlar al menos parcialmente si la estructura de sintaxis de capa de MB incluye un elemento de sintaxis transform_size_8x8_flag. El proceso de codificación CABAC para el indicador vsp_direct puede ser el mismo que el proceso de codificación CABAC para mb_direct_type_flag en el Borrador de Trabajo 1 3D-AVC.

[0189] En el ejemplo de la Tabla 11, la estructura de sintaxis de predicción de MB puede incluir mb_part_vsp_flag [mbPartIdx] si slice_vsp_flag es igual a 1 y VspRefExist es igual a 1. En otros ejemplos, la estructura de sintaxis de predicción de MB puede incluir mb_part_vsp_flag [mbPartIdx] si slice_vsp_flag es igual a 1.

[0190] Una séptima implementación de ejemplo de las técnicas de esta divulgación proporciona un elemento de sintaxis combinado que indica si un MB se codifica en modo de omisión y, si es así, si el MB se va a decodificar basándose en una imagen VSP. En la séptima implementación de ejemplo, las estructuras sintácticas de datos de fragmento pueden conformarse a la sintaxis de la Tabla 12, a continuación.

TABLA 12 - Sintaxis de datos de fragmento

slice_data() {	C	Descriptor
if(entropy_coding_mode_ _flag)		
while(!byte_aligned())		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data()		
} else {		
if(slice _vsp_enable && VspRefExist) {		
mb_skip_idc	2	ae(v)
moreDataFlag = (mb_skip_idc>1)		
}else{		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
}		
if(moreDataFlag) {		
if(MbaffFrameFlag && (CurrMbAddr % 2 = = 0 || (CurrMbAddr % 2 = = 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) | ae(v)
macroblock_layer( )	2 | 3 | 4	
}		
if(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data()		
else {		
if(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
if(MbaffFrameFlag && CurrMbAddr % 2 = = 0)		
moreDataFlag = 1		
else {		

slice_data() {	C	Descriptor
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} while(moreDataFlag)		
}		

[0191] En la sintaxis de ejemplo de la Tabla 12, una estructura de sintaxis de datos de fragmento para un fragmento puede incluir un elemento de sintaxis mb_skip_idc si la VSP está habilitada para el fragmento y existe una imagen de referencia de VSP para el fragmento. En un ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 0, el MB actual se omite de una imagen VSP. Esto significa que no se señala ningún movimiento ni residuo para el MB actual. El MB actual es el MB al que se aplica el elemento de sintaxis mb_skip_idc. En este ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 1, el MB actual se codifica usando el modo de omisión normal. En otras palabras, el MB actual no se omite de una imagen VSP. Además, en este ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 2, el MB actual no se codifica usando el modo de omisión.

[0192] En algunos ejemplos, el elemento de sintaxis mb_skip_idc puede tener valores adicionales. Por ejemplo, en un ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 2, el MB actual se predice totalmente a partir de una imagen VSP y, por tanto, no se indica información de movimiento para el MB actual, pero puede señalarse información de residuos para el MB actual. En otro ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 3, el MB actual se predice totalmente desde un modo de predicción directa normal, mientras que el residuo del MB se señala. El modo de predicción directa es un tipo de interpredicción en el que no se señala ni decodifica ningún vector de movimiento. En el modo de predicción directa, se señala un índice de referencia de una imagen de referencia y se señala un residuo del MB (o partición de MB) en relación con un MB (o partición de MB) situado de la imagen de referencia. En otro ejemplo, un elemento de sintaxis asociado con un MB puede indicar que todo el MB se predice a partir de la imagen VSP y que se señala un residuo para el MB. Es decir, similar a otras técnicas de esta divulgación, el modo adicional de indicar toda una MV se predice a partir de la VSP pero el residuo que se sigue transmitiendo se puede señalar en el MB.

[0193] En otro ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 1, el MB actual se omite de una imagen VSP. En este ejemplo, no se indica ninguna información de movimiento ni residuo para el MB actual. Además, en este ejemplo, si el elemento de sintaxis mb_skip_idc es igual a 0, el MB actual se codifica en un modo de omisión normal.

[0194] Además, las estructuras de sintaxis de capa de MB para el MB pueden conformarse a la sintaxis de la Tabla 13, a continuación.

TABLA 13 - Sintaxis de capa de macrobloque

macroblock_layer() {	C	Descriptor
if (slice_vsp_flag && VspRefExist)		
vsp_mb_flag	2	ae(v)
if (!vsp_mb_flag)		
mb_type	2	ue(v) | ae(v)
if(mb_type = = I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	3	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i]	3	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i]	3	u(v)

macroblock_layer() {	C	Descriptor
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		

if(mb_type != I_NxN &&
MbPartPredMode(mb_type, 0) != Intra_16x16 &&
NumMbPart(mb_type) = = 4) {		

if (!vsp_mb_flag)		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type == I_NxN)		
transform_size_8x8_flag	2	u(1) | ae(v)
if (!vsp_mb_flag)		
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_ 16x16) {		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (vsp_mb_flag || (!vsp_mb_flag && (mb_type != B_Direct_16x16 || direct 8x8 inference_ flag))))		
transform_size_8x8_flag	2	u(1) | ae(v)
}		
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_ type, 0) = = Intra 16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual(0, 15)	3 | 4	
}		
}		
}		

[0195] La estructura de sintaxis de capa de MB de ejemplo de la Tabla 13 puede incluir un elemento de sintaxis vsp_mb_flag si la VSP está habilitada para el fragmento y existe una imagen de referencia de la VSP para el fragmento. El elemento de sintaxis vsp_mb_flag indica si el MB completo correspondiente a la estructura de sintaxis de capa de MB se predice a partir de una imagen VSP y si se señalan los datos residuales para el MB. Por ejemplo, si el elemento de sintaxis vsp_mb_flag es igual a 1, se puede predecir todo el MB a partir de una imagen VSP y se señalan los datos residuales para el MB. En este ejemplo, si el elemento de sintaxis vsp_mb_flag es igual a 0, todo el MB puede predecirse por otros modos de codificación. Además, en este ejemplo, cuando el elemento de sintaxis vsp_mb_flag es igual a 1, el elemento de sintaxis mb_type no se señala en la estructura de sintaxis de capa de MB. Cuando el

elemento de sintaxis vsp_mb_flag no está presente en la estructura de sintaxis de capa de MB, el decodificador de vídeo 30 puede deducir que el elemento de sintaxis vsp_mb_flag sea igual a 0.

[0196] En las estructuras de sintaxis de ejemplo descritas en las Tablas 12 y 13, el modelado de contexto para elementos de sintaxis distintos del elemento de sintaxis mb_skip_idc puede ser el mismo que se describe en otra parte en esta divulgación. Las probabilidades iniciales para los contextos del elemento de sintaxis mb_skip_idc se pueden inicializar con diferentes valores para los fragmentos P y los fragmentos B. En diversos ejemplos, el elemento de sintaxis mb_skip_idc se puede binarizar de diversas maneras y se puede seleccionar un contexto para el elemento de sintaxis mb_skip_idc de diversas maneras.

[0197] En un ejemplo, el elemento de sintaxis mb_skip_idc se puede binarizar usando información de un MB denominado mbPAddrA y un MB denominado mbPAddrB. mbPAddrA puede ser un MB vecino (o un bloque 8x8) a la izquierda del MB actual o de la partición de MB actual. mbPAddrB puede ser un MB vecino (o un bloque 8x8) encima del MB actual o de la partición de MB actual. Si tanto mbPAddrA como mbPAddrB están disponibles y tanto mbPAddrA como mbPAddrB se codifican usando el modo de omisión de la VSP, el elemento de sintaxis mb_skip_idc se puede binarizar de acuerdo con la Tabla 14 siguiente:

Tabla14 - Binarización de mb_skip_idx

Valor (nombre) de mb_skip_idc	**Secuencia de bins**						
0 (modo de omisión de VSP)	1						
1 (modo de omisión normal)	0	0					
2 (modos sin omisión)	0	1					
Índice de bin (binIdx)	*0*	*1*	*2*	*3*	*4*	*5*	*6*

De forma alternativa, el elemento de sintaxis mb_skip_idc se puede binarizar de acuerdo con la Tabla 15 siguiente:

TABLA 15 - Binarización de mb_skip_idx

Valor (nombre) de mb_skip_idc	**Secuencia de bins**						
0 (modo de omisión de VSP)	1						
1 (modo de omisión normal)	0	1					
2 (modos sin omisión)	0	0					
Índice de bin (binIdx)	*0*	*1*	*2*	*3*	*4*	*5*	*6*

[0198] Cuando el codificador por entropía está codificando por entropía un elemento de sintaxis mb_skip_idx binarizado de acuerdo con la Tabla 14 o la Tabla 15, el codificador por entropía puede determinar un ctxIdxInc para cada bin del elemento de sintaxis mb_skip_idx binarizado. En este ejemplo, si binIdx de un bin del elemento de sintaxis mb_skip_idx binarizado es igual a 0 (es decir, el codificador por entropía está codificando por entropía el primer bin del elemento de sintaxis mb_skip_idx), el ctxIdxInc para el bin puede ser igual a la suma de condTermFlagA y condTermFlagB. Es decir, ctxIdxInc = condTermFlagA + condTermFlagB. En este ejemplo, condTermFlagN (donde N es A o B) puede ser igual a 0 si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión de VSP. De lo contrario, condTermFlagN puede ser igual a 1.

[0199] Además, en este ejemplo, si el binIdx de un bin es igual a 1, el ctxIdxInc para el bin puede ser igual a condTermFlagA más condTermFlagB. Cuando el binIdx del bin es igual a 1, condTermFlagN (donde N es A o B) puede ser igual a 0 si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión normal. De lo contrario, condTermFlagN puede ser igual a 1.

[0200] De lo contrario, si mbPAddrA o mbPAddrB no están disponibles o si mbPAddrA o mbPAddrB no se codifican en el modo de omisión de VSP, el elemento de sintaxis mb_skip_idc se puede binarizar de acuerdo con la Tabla 16, a continuación:

TABLA 16 - Binarización de mb_skip_idc

Valor (nombre) de mb_skip_idc	**Secuencia de bins**						
0 (modo de omisión normal)	1						
1 (modo de omisión de VSP)	0	0					
2 (modos sin omisión)	0	1					
Índice de bin (binIdx)	*0*	*1*	*2*	*3*	*4*	*5*	*6*

De forma alternativa, si mbPAddrA o mbPAddrB no está disponible o si mbPAddrA o mbPAddrB no se codifica en el modo de omisión de VSP, el elemento de sintaxis mb_skip_idc se puede binarizar de acuerdo con la Tabla 17 siguiente:

TABLA 17 - Binarización de mb_skip_idc

Valor (nombre) de mb_skip_idc	**Secuencia de bins**						
0 (modo de omisión normal)	1						
1 (modo de omisión de VSP)	0	1					
2 (modos sin omisión)	0	0					
Índice de bin (binIdx)	*0*	*1*	*2*	*3*	*4*	*5*	*6*

[0201] Cuando mbPAddrA o mbPAddrB no está disponible o mbPAddrA o mbPAddrB no se codifica en el modo de omisión de VSP, el codificador por entropía puede determinar un ctxIdxInc para cada bin del elemento de sintaxis mb_skip_idx binarizado. En este ejemplo, si binIdx de un bin del elemento de sintaxis mb_skip_idx binarizado es igual a 0 (es decir, el codificador por entropía está codificando por entropía el primer bin del elemento de sintaxis mb_skip_idx), el ctxIdxInc para el bin puede ser igual a la suma de condTermFlagA y condTermFlagB. Es decir, ctxIdxInc = condTermFlagA + condTermFlagB. En este ejemplo, condTermFlagN (donde N es A o B) puede ser igual a 0 si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión normal. De lo contrario, condTermFlagN puede ser igual a 1.

[0202] Además, en este ejemplo, si el binIdx de un bin es igual a 1, el ctxIdxInc para el bin puede ser igual a condTermFlagA más condTermFlagB. Cuando el binIdx del bin es igual a 1, condTermFlagN (donde N es A o B) puede ser igual a 0 si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión de VSP. De lo contrario, condTermFlagN puede ser igual a 1.

[0203] En otros ejemplos, el ctxIdxInc de un bin puede depender de un único vecino espacial. Por ejemplo, el ctxIdxInc puede ser igual a condTermFlagA. En otro ejemplo, el ctxIdxInc puede ser igual a condTermFlagB. En este ejemplo, condTermFlagA y condTermFlagB pueden definirse como anteriormente.

[0204] Una octava implementación de ejemplo de las técnicas de esta divulgación es similar a la séptima implementación de ejemplo, sin embargo, la binarización y los procedimientos de selección de contexto son diferentes. En la octava implementación de ejemplo, los esquemas de binarización para MB P en los fragmentos P y para MB B en los fragmentos B se especifican de acuerdo con la Tabla 18 siguiente:

TABLA 18 - Binarización de mb_skip_idc

Valor (nombre) de mb_skip_idc	**Secuencia de bins**						
0 (modo de omisión normal/modo de omisión de VSP)	0	0					
1 (modo de omisión de VSP/modo de omisión normal)	0	1					
2 (modos sin omisión)	1						
Índice de bin (binIdx)	*0*	*1*	*2*	*3*	*4*	*5*	*6*

[0205] De forma alternativa, en la octava implementación de ejemplo, el proceso de binarización para MB P en los fragmentos P y para MB B en los fragmentos B se especifica de acuerdo con la Tabla 19 siguiente:

TABLA 19 - Binarización de mb_skip_idc

Valor (nombre) de mb_skip_idc	**Secuencia de bins**						
0 (modo de omisión normal/modo de omisión de VSP)	1	0					
1 (modo de omisión de VSP/modo de omisión normal)	1	1					
2 (modos sin omisión)	0						
Índice de bin (binIdx)	*0*	*1*	*2*	*3*	*4*	*5*	*6*

[0206] En la octava implementación de ejemplo, cuando un codificador por entropía esté codificando un elemento de sintaxis MB_skip_idc, el codificador por entropía puede determinar un ctxIdxInc para cada bin del elemento de sintaxis mb_skip_idc binarizado. Por ejemplo, cuando se decodifique mb_skip_idc, se puede especificar el ctxIdxInc para mb_skip_idc dependiendo del binIdx. Al determinar el ctxIdxInc para un bin, mbPAddrA puede indicar el MB vecino (o el bloque 8x8) a la izquierda del MB actual y mbPAddrB puede indicar el MB vecino (o el bloque 8x8) encima del MB o de la partición de MB actual.

[0207] Si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es igual a 0, un codificador por entropía puede determinar el ctxIdxInc para el bin de diversas maneras. Por ejemplo, si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es igual a 0, el ctxIdxInc para el bin puede ser igual a la suma de condTermFlagA y condTermFlagB. En este ejemplo, condTermFlagN (donde N es A o B) puede ser igual a 0 si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión (ya sea el modo de omisión normal o el modo de omisión de VSP). De lo contrario, en este ejemplo, condTermFlagN es igual a 1. En otro ejemplo, si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es igual a 0, el ctxIdxInc para el bin puede ser igual a la suma de condTermFlagA y condTermFlagB. En este ejemplo, condTermFlagN (donde N es A o B) puede ser igual a 0 si mbPAddrN no está disponible o el modo de mbPAddrN no es el modo de omisión (ni el modo de omisión normal ni el modo de omisión de VSP). De lo contrario, en este ejemplo, condTermFlagN es igual a 1. En otro ejemplo, si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es igual a 0, el ctxIdxInc para el bin puede ser igual a condTermFlagN, donde condTermFlagN (donde N es A o B) se define de la manera descrita por los otros ejemplos de este párrafo.

[0208] Si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es mayor que 0, el codificador por entropía puede determinar el ctxIdxInc para el bin de diversas maneras. Por ejemplo, si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es mayor que 0, el ctxIdxInc para el bin puede ser igual a la suma de condTermA, condTermB y ctxOffset. En este ejemplo, condTermFlagN (donde N puede ser A o B) y ctxOffset pueden depender del modo de mbPAddrN. Además, en este ejemplo, el ctxOffset puede ser igual a 0 si tanto mbPAddrA como mbPAddrB están disponibles y codificados como modo de omisión de VSP. De lo contrario, el ctxOffset puede ser igual a 3. En uno, de forma alternativa, el ctxOffset se establece en 0. En otra alternativa, si tanto mbPAddrA como mbPAddrB están disponibles y codificados como modo de omisión de VSP, el ctxOffset es igual a 3 e igual a 0 en caso contrario. En este ejemplo, si tanto mbPAddrA como mbPAddrB están disponibles yse codifican como modo de omisión de VSP, condTermFlagA y condTermFlagB pueden establecerse igual a 0. De forma alternativa, si mbPAddrN (donde N puede ser A o B) no está disponible o el modo de mbPAddrN es el modo de omisión de VSP, condTermFlagN se establece igual a 0 y se establece igual a 1 en caso contrario.

[0209] Continuando con el ejemplo del párrafo anterior, si mbPAddrA o mbPAddrB no está disponible o no se codifica como modo de omisión de VSP, condTermFlagN (donde N es A o B) puede establecerse igual a 0. De lo contrario, condTermFlagN se puede configurar igual a 1. En un ejemplo alternativo, si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión de VSP, condTermFlagN se establece igual a 0 y se establece igual a 1 en caso contrario. En otro ejemplo alternativo, si mbPAddrN no está disponible o el modo de mbPAddrN no es el modo de omisión de VSP, condTermFlagN se puede configurar igual a 0 y se puede establecer igual a 1 en caso contrario. En otra alternativa, si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión normal, condTermFlagN se puede establecer en 0. De lo contrario, condTermFlagN se puede configurar igual a 0. En otra alternativa, si mbPAddrN no está disponible o el modo de mbPAddrN no es el modo de omisión normal, condTermFlagN puede establecerse igual a 0. De lo contrario, condTermFlagN puede establecerse en 1.

[0210] En otro ejemplo de acuerdo con la octava implementación de ejemplo, si el binIdx de un bin de un elemento de sintaxis mb_skip_idc binarizado es mayor que 0, el codificador por entropía puede determinar ctxIdxInc para el bin como la suma de condTermFlagN y ctxOffset. En este ejemplo, condTermFlagN (donde N puede ser A o B) y el ctxOffset pueden depender del modo de mbPAddrN. Además, en este ejemplo, ctxOffset puede establecerse en 0 si mbPAddrN está disponible y se codifica como modo de omisión de VSP. De lo contrario, ctxOffset puede establecerse en 2. En una alternativa, ctxOffset se establece en 0. En otra alternativa, si mbPAddrN está disponible y codificado como modo de omisión de VSP, el ctxOffset se establece en 2. De lo contrario, en esta alternativa, el ctxOffset se establece en 0.

[0211] Continuando con el ejemplo del párrafo anterior, si mbPAddrN está disponible y se codifica como modo de omisión de VSP, condTermFlagN se puede definir de la siguiente manera: si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión normal, condTermFlagN se establece igual a 0,y, de lo contrario, condTermFlagN se establece en 1. De lo contrario, si mbPAddrN no está disponible o no se codifica como modo de omisión de VSP, condTermFlagN puede definirse como: si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión normal, condTermFlagN se establece igual a 0 y se establece igual a 1. En una alternativa, si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión de VSP, condTermFlagN se establece igual a 0. De lo contrario, condTermFlagN se establece igual a 1. En otra alternativa, si mbPAddrN no está disponible o el modo de mbPAddrN no es el modo de omisión de VSP, condTermSkipFlagN se establece igual a 0. De lo contrario, se establece en 1. En otra alternativa, si mbPAddrN no está disponible o el modo de mbPAddrN es el modo de omisión normal, condTermFlagN se establece igual a 0. De lo contrario, condTermFlagN se establece igual a 0. En otra alternativa, si mbPAddrN no está disponible o el modo de mbPAddrN no es el modo de omisión normal, condTermSkip-FlagN se establece igual a 0. De lo contrario, se establece en 1.

[0212] Como se menciona anteriormente, algunas técnicas de esta divulgación son compatibles con múltiples imágenes VSP. Dichas técnicas pueden ser adicionales a la séptima u octava implementación (u otras implementaciones) descritas en esta divulgación. En dichas técnicas, el número total de imágenes VSP disponibles puede señalarse en diversas partes de un flujo de bits. Por ejemplo, el número total de imágenes VSP disponibles puede señalarse en un conjunto de parámetros de secuencia, en un conjunto de parámetros de imagen, en una cabecera de fragmento o en otra parte del flujo de bits. Cuando haya múltiples imágenes VSP disponibles, se puede señalar un elemento de sintaxis para indicar un índice de una imagen VSP aplicable. El siguiente análisis puede aplicarse cuando el número de imágenes VSP sea mayor que 1. La Tabla 20 siguiente muestra una estructura de sintaxis de ejemplo para los datos de fragmento. La estructura de sintaxis de la Tabla 20 incluye un elemento de sintaxis vsp ref idx que especifica un índice de una imagen VSP aplicable a un MB.

TABLA 20 - Sintaxis de datos de fragmento

slice_data () {	C	Descriptor
if(entropy_coding_mode_flag)		
while(!byte_aligned())		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data()		
} else {		
if(slice_vsp_enable && VspRefExist) {		
mb_skip_idc	2	ae(v)
if(MbVspSkip)		
vsp_ref_idx	2	ae(v)
moreDataFlag = (mb_skip_idc>1)		
}else{		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
}		
if(moreDataFlag) {		

slice_data () {	C	Descriptor
if(MbaffFrameFlag && (CurrMbAddr % 2 = = 0 || (CurrMbAddr % 2 = = 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) | ae(v)
macroblock_layer( )	2 | 3 | 4	
}		
if(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data()		
else {		
if(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
if(MbaffFrameFlag && CurrMbAddr % 2 = = 0)		
moreDataFlag = 1		
else {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} while(moreDataFlag)		
}		

[0213] En la estructura de sintaxis de ejemplo de la Tabla 20, el elemento de sintaxis vsp_ref_idx puede especificar el índice de la imagen VSP que se usará para la predicción. La estructura de sintaxis de datos de fragmento puede incluir el elemento de sintaxis vsp_ref_idx si el MB actual se omite de una imagen VSP que se puede derivar de la imagen VSP indicada por el elemento de sintaxis mb_skip_idc. Además, la estructura de sintaxis de datos de sectores puede incluir el elemento de sintaxis vsp_ref_idx si el elemento de sintaxis vsp_mb_flag es igual a 1. Además, la estructura de sintaxis de datos de fragmento puede incluir el elemento de sintaxis vsp_mb_flag si mb_part_vsp_flag [mbPartIdx] es igual a 1. Como se analizó anteriormente, mb_part_vsp_flag [mbPartIdx] indica si la partición de MB actual, que se identifica mbPartIdx, se predice a partir de una imagen VSP. La estructura de sintaxis de datos de fragmento también puede incluir el elemento de sintaxis vsp_mb_flag si sub_mb_vsp_flag[mbPartIdx] es igual a 1. Como se analizó anteriormente, sub_mb_vsp_flag[mbPartIdx] indica si la partición de MB actual (8x8), que se identifica como mbPartIdx, se predice a partir de una imagen VSP.

[0214] La Tabla 21 siguiente muestra una sintaxis de ejemplo para una estructura de sintaxis de capa de MB. Como se muestra en la Tabla 21, una estructura de sintaxis de capa de MB puede incluir un elemento de sintaxis vsp_ref_idx que identifique una imagen VSP entre una pluralidad de imágenes VSP disponibles.

TABLA 21 - Sintaxis de capa de macrobloque

macroblock_layer() {	C	Descriptor
if (slice_vsp_flag && VspRefExist) {		
vsp_mb_flag	2	ae(v)
if(vsp_mb_flag)		
vsp_ref_idx	2	ae(v)
}		
if (! vsp_mb_flag)		
mb_type	2	ue(v) | ae(v)
if(mb_type = = I_PCM){		
while(!byte_aligned())		
pcm_alignment_zero_bit	3	f(1)
for(i = 0; i < 256; i++)		

macroblock_layer() {	C	Descriptor
pcm_sample_luma[i]	3	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i]	3	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		
if(mb_type != I_NxN && MbPartPredMode(mb_type, 0) != Intra_16x16 &&		
NumMbPart(mb_type) = = 4) {		
if (!vsp_mb_flag)		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type = = I_NxN)		
transform_size_8x8_flag	2	u(1) | ae(v)
if (!vsp_mb_flag)		
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_16x16) {		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (vsp_mb_flag || (!vsp_mb_flag && (mb_type != B_Direct_16x16 || direct 8x8 inference flag))))		
transform_size_8x8_flag	2	u(1) | ae(v)
}		
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) = = Intra 16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual(0, 15)	3 | 4	
}		
}		
}		

[0215] En la Tabla 21, la semántica del elemento de sintaxis vsp_ref_idx puede ser la misma que la descrita anteriormente con referencia a la Tabla 20 y la estructura de sintaxis de capa de MB puede incluir el elemento de sintaxis vsp_ref_idx cuando se produzcan las mismas condiciones.

[0216] Además, la Tabla 22, a continuación, muestra una sintaxis de ejemplo para una estructura de sintaxis de predicción de MB. Como se muestra en la Tabla 22, una estructura de sintaxis de predicción de MB puede incluir un elemento de sintaxis vsp_ref_idx[mbPartIdx] que identifique, para una partición de MB identificada por mbPartIdx, una imagen VSP entre una pluralidad de imágenes VSP disponibles.

TABLA 22 - Sintaxis de predicción de macrobloque

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) = = Intra 4x4 ||		
MbPartPredMode(mb_type, 0) = = Intra 16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag[luma4x4BlkIdx])		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if (slice _vsp_flag && VspRefExist) //vsp_mb_flag is not 1		
mb_part_vsp_flag[mbPartIdx **]**	2	ae(v)
if(mb_part_vsp_flag[mbPartIdx])		
vsp_ref_idx[mbPartIdx]	2	ae(v)
if(!mb_part_vsp_flag[mbPartIdx] &&(num_ref_idx_10_active_minus1 > 0 || mb_field_decoding_fLag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx-l0[mbPartIdx **]**	2	te(v) | ae(v)
for(!mb_part_vsp_flag[mbPartIdx]&&(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag[mbPartIdx] &&(MbPartPredMode (mb type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][**0**][**compIdx**]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type_); mbPartIdx++)		
if(!mb_part_vsp_flag[mbPartIdx] &&(MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

[0217] En la Tabla 22, la semántica del elemento de sintaxis vsp_ref_idx puede ser la misma que la descrita anteriormente con referencia a la Tabla 20, y la estructura de sintaxis de predicción de MB puede incluir el elemento de sintaxis vsp_ref_idx cuando se produzcan las mismas condiciones.

[0218] La Tabla 23 siguiente muestra una sintaxis de ejemplo para una estructura de sintaxis de predicción de subMB. Como se muestra en la Tabla 23, una estructura de sintaxis de predicción de subMB puede incluir un elemento de sintaxis vsp_ref_idx [mbPartIdx] que identifique, para una partición de subMB identificada por mbPartIdx, una imagen VSP entre una pluralidad de imágenes VSP disponibles.

TABLA 23 - Sintaxis de predicción de submacrobloque

sub_mb_pred (mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++) {		
if (slice_vsp_flag && VspRefExist) // vsp_mb_flag is not 1		
sub_mb_vsp_flag[mbPartIdx **]**		
if(sub_mb_vsp_flag**[** mbPartIdx **]**)		
vsp_ref_idx[mbPartIdx]	2	ae(v)
else		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
}		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[**mbPartIdx**]**) &&(num_ref_idx_10_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v)| ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag[mbPartIdx]) && (num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v)| ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[** mbPartIdx **]**) &&(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub _mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag**[** mbPartIdx **]**) &&sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub _mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

[0219] En la Tabla 23, la semántica del elemento de sintaxis vsp_ref_idx puede ser la misma que la descrita anteriormente con referencia a la Tabla 20, y la estructura de sintaxis de predicción de MB puede incluir el elemento de sintaxis vsp_ref_idx cuando se produzcan las mismas condiciones.

[0220] El proceso de binarización para vsp_ref_idx puede ser el mismo que ref_idx_l0 y ref_idx_l1. La selección de contexto de vsp_ref_idx se puede ejecutar de forma similar a ref_idx_l0 y ref idx_l1. La norma H.264/AVC describe el proceso de binarización y la selección de contexto para ref idx_l0 y ref_idx_l1. El modelado de contexto para vsp_ref_idx se puede basar en los valores de vsp_ref_idx de las particiones de MB o de los MB vecinos, pero no de los valores de ref_idx_l0 o ref_idx_l1.

[0221] En algunos ejemplos, se puede predecir un MB o una partición de MB basándose en dos imágenes VSP. En algunos de dichos ejemplos, las estructuras de sintaxis de predicción de MB pueden tener la sintaxis descrita en la Tabla 24 siguiente.

TABLA 24 - Sintaxis de predicción de macrobloque

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_type, 0) = = Intra_16x16){		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(slice_vsp_flag && NumMbPart(mb_type) != 1 && VspRefExist) {		
if (MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
mb_part_vsp_flag_L0[mbPartIdx **]**	2	ae(v)
if (MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
mb_part_vsp_flag_L1[mbPartIdx **]**	2	ae(v)
}		
if(!mb_part_vsp_flag_L0[mbPartIdx] &&(num_ref_idx_10_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for (mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type) && !mb_part_vsp_flag_L1[mbPartIdx]; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag_L0**[**mbPartIdx **]**&& MbPartPredMode (mb_type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][**0**][**compIdx**]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag_L1 **[** mbPartIdx **]** && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

[0222] En la sintaxis de ejemplo de la Tabla 24, la estructura de sintaxis de predicción de MB puede incluir un elemento de sintaxis mb_part_vsp_flag_l0[mb-PartIdx] y un elemento de sintaxis mb_part_vsp_flagl1[mbPartIdx]. El elemento de sintaxis mb_part_vsp_flag_10[mbPartIdx] indica si la partición de MB actual se predice a partir de una imagen de referencia normal en RefPicListO o de una imagen VSP. Por ejemplo, si mb_part_vsp_flag_10[mbPartIdx] es igual a 1, la partición de MB actual no se predice a partir de una imagen normal en RefPicListO para la predicción, sino a partir de una imagen VSP. En este ejemplo, si mb_part_vsp_flag_10[mb-PartIdx] es igual a 0, la partición de MB se predice a partir de una imagen normal en RefPicListO para la predicción. De forma similar, el elemento de sintaxis

mb_part_vsp_flag_l1[mbPartIdx] indica si la partición de MB actual se predice a partir de una imagen de referencia normal en RefPicList1 o de una imagen VSP. Por ejemplo, si mb_part_vsp_flag_l1[mbPartIdx] es igual a 1, la partición de MB actual no se predice a partir de una imagen normal en RefPicList para la predicción, sino a partir de una imagen VSP. En este ejemplo, si el indicador mb_part_vsp_flag_l1[mbPartIdx] es igual a 0, la partición de MB se predice a partir de una imagen normal en RefPicList1 para la predicción.

[0223] Además, algunos ejemplos en los que un MB o una partición de MB puede predecirse basándose en dos imágenes VSP, las estructuras de sintaxis de predicción de subMB pueden tener la sintaxis descrita en la Tabla 25 siguiente.

TABLA 25 - Sintaxis de predicción de submacrobloque

sub_mb_pred(mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++) {		
if (slice_vsp_flag && VspRefExist) // vsp_mb_flag is not 1		
sub_mb_vsp_ftag[mbPartIdx**]**		
if (!sub_mb_vsp_flag**[**mbPartIdx **]**)		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
if (slice_vsp_flag && VspRefExist && (NumSubMbPart (sub_mb_type[mbPartIdx]) == 1 || sub_mb_type[mbPartIdx] = = B_Direct_8x8)) {		
if(SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
sub_mb_vsp_flag_l0[mbPartIdx **]**	2	ae(v)
if(SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
sub_mb_vsp_flag_l1[mbPartIdx **]**	2	ae(v)
}		
}		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l0**[**mbPartIdx**]** &&(num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l1**[** mbPartIdx **]** && (num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l0**[** mbPartIdx **]** &&(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type [mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l1 **[** mbPartIdx **]** &&sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		

mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

[0224] En la sintaxis de ejemplo de la Tabla 25, una estructura de sintaxis de predicción de subMB puede incluir un elemento de sintaxis sub_mb_vsp_flag[mb-PartIdx], un elemento de sintaxis sub_mb_vsp_flag_l0[mbPartIdx] y un elemento de sintaxis sub_mb_vsp_flag_l1[mbPartIdx]. sub_mb_vsp_flag[mbPartIdx] indica si la partición de MB actual (8x8) se predice a partir de una imagen VSP. Por ejemplo, si sub_mb_vsp_flag[mbPartIdx] es igual a 1, la partición de MB actual (8x8) se predice a partir de una imagen VSP. En este ejemplo, si sub_mb_vsp_flag[mbPartIdx] es igual a 0, la partición de MB actual (8x8) no se predice a partir de una imagen VSP. Cuando sub_mb_vsp_flag[mbPartIdx] no esté presente, se puede deducir que sub_mb_vsp_flag[mbPartIdx] es igual a 0. Cuando sub_mb_vsp_flag[mbPartIdx] es igual a 1, tanto mb_part_vsp_flag_l0[mbPartIdx] como mb_part_vsp_flag_l1[mbPartIdx] se derivan para que sean igual a 1.

[0225] sub_mb_vsp_flag_l0[mbPartIdx] indica si la partición de MB actual se predice a partir de una imagen de referencia normal en RefPicListO o de una imagen VSP. Por ejemplo, si sub_mb_vsp_flag_l0[mbPartIdx] es igual a 1, la partición de MB actual no se predice a partir de una imagen normal en RefPicList0 para la predicción, sino a partir de una imagen VSP. En este ejemplo, si sub_mb_vsp_flag_l0[mbPartIdx] es igual a 0, la partición de MB se predice a partir de una imagen normal en RefPicList0 para la predicción. De forma similar, el elemento de sintaxis sub_mb_vsp_flag_l1[mbPartIdx] indica si la partición de MB actual se predice a partir de una imagen de referencia normal en RefPicList1 o de una imagen VSP. Por ejemplo, si sub_mb_vsp_flag_l1[mbPartIdx] es igual a 1, la partición de MB actual no se predice a partir de una imagen normal en RefPicList1 para la predicción, sino a partir de una imagen VSP. En este ejemplo, si el indicador sub_mb_vsp_flag_l1[mbPartIdx] es igual a 0, la partición de MB se predice a partir de una imagen normal en RefPicList1 para la predicción.

[0226] En algunos ejemplos, la imagen VSP se incluye en una lista de imágenes de referencia y ref_idx se usa para derivar los valores de sub_mb_vsp_flagl0, sub_mb_vsp_flagl1, mb_part_vsp_flag_l0 y mb_part_vsp_flag_l1. En uno de dichos ejemplos, la imagen VSP puede incluirse como la entrada ref_idx_l0_vsp de RefPicListO. En este ejemplo, ref_idx_l0_vsp indica una posición en RefPicListO de la imagen VSP. Además, en este ejemplo, la imagen VSP se puede incluir como la entrada ref_idx_l1_vsp de RefPicList1. En otras palabras, ref_idx_l1_vsp indica una posición en RefPicList1 de la imagen VSP. Si no hay una imagen VSP en RefPicList0, ref_idx_l0_vsp es igual a -1. Si no hay una imagen VSP en RefPicList1, ref_idx_l1_vsp es igual a -1. Esto se muestra en los Tablas 26 y 27, a continuación.

TABLA 26 - Sintaxis de predicción de macrobloque

mb_pred(mb_type) {	**C**	**Descriptor**
if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1) {		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
mb_part_vsp_flag_L0[mbPartIdx]= (ref_idx_l0[mbPartIdx] = = ref_idx_l0_vsp)		
}		
for (mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0) {		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)

mb_part_vsp_flag_L1[mbPartIdx]= (ref idx_l1[mbPartIdx] = = ref_idx_l1_vsp)		
}		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag_L0**[** mbPartIdx **]** && MbPartPredMode (mb_type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(!mb_part_vsp_flag_L1 **[** mbPartIdx **]** && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][**0**][**compIdx**]**	2	se(v) | ae(v)
}		
}		

[0227] Como se muestra en la Tabla 26, mb_part_vsp_flag_L0[mbPartIdx] puede ser igual a 1 si ref_idx_l0[mbPartIdx] es igual a ref_idx_l0_vsp. Como se indicó anteriormente, mb_part_vsp_flag_L0[mbPartIdx] igual a 1 puede indicar que la partición de MB actual no se predice a partir de una imagen normal en RefPicList0, sino que se predice a partir de una imagen VSP. De forma similar, el decodificador de vídeo 30 puede determinar que mb_part_vsp_flag_L1[mbPartIdx] sea igual a 1 si ref_idx_l1[mbPartIdx] es igual a ref_idx_l1_vsp. mb_part_vsp_flag_L0[mbPartIdx] igual a 1 puede indicar que la partición de MB actual no se predice a partir de una imagen normal en RefPicList1, sino que se predice a partir de una imagen VSP. Por tanto, en la sintaxis de ejemplo de la Tabla 26, puede ser innecesario señalar un elemento de sintaxis separado para indicar si la partición de MB actual se predice a partir de una imagen normal o de una imagen VSP. En su lugar, el elemento de sintaxis ref_idx_l0 se puede reutilizar con el fin de determinar si la partición de MB actual se predice a partir de una imagen normal o de una imagen VSP.

TABLA 27 - Sintaxis de predicción de submacrobloque

sub_mb_pred(mb_type) {	**C**	**Descriptor**
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if (slice_vsp_flag && VspRefExist) // vsp_mb_flag is not 1		
sub_mb_vsp_flag[mbPartIdx **]**		
if (!sub_mb_vsp_flag**[** mbPartIdx **]**)		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1) {		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
sub_mb_vsp_flag_l0**[** mbPartIdx **]** = (ref_idx_l0[mbPartIdx] = = ref_idx_l0_vsp)		
}		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l1 **[**mbPartIdx**]** && (num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0) {		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
sub_mb_vsp_flag_l1 **[**mbPartIdx**]** = (ref_idx_l1[mbPartIdx] = = ref_idx_l1_vsp)		
}		
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l0**[** mbPartIdx **]** &&(sub_mb_type[mbPartIdx] !=		

sub_mb_pred(mb_type) {	C	Descriptor
B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type [mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx**][** subMbPartIdx**][**compIdx**]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(!sub_mb_vsp_flag_l1**[** mbPartIdx **]** &&sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		

for(subMbPartIdx = 0;
subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]);
subMbPartIdx++)		

for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

[0228] Como se muestra en la Tabla 27, el decodificador de vídeo 30 puede determinar que sub_mb_vsp_flag_L0[mbPartIdx] es igual a 1 si ref_idx_l0[mbPartIdx] es igual a ref_idx_l0_vsp. Como se indicó anteriormente, sub_mb_vsp_flag_L0[mbPartIdx] igual a 1 puede indicar que la partición de MB actual (8x8) no se predice a partir de una imagen normal en RefPicList0, sino que se predice a partir de una imagen VSP. De forma similar, el decodificador de vídeo 30 puede determinar que sub_mb_vsp_flag_L1[mbPartIdx] es igual a 1 si ref_idx_l1[mbPartIdx] es igual a ref_idx_l1_vsp. sub_mb_vsp_flag_L0[mbPartIdx] igual a 1 puede indicar que la partición de MB actual (8x8) no se predice a partir de una imagen normal en RefPicList1, sino que se predice a partir de una imagen VSP. Por tanto, en la sintaxis de ejemplo de la Tabla 27, puede ser innecesario señalar un elemento de sintaxis separado para indicar si la partición de MB actual (8x8) se predice a partir de una imagen normal o de una imagen VSP. En su lugar, el elemento de sintaxis ref_idx_l0 se puede volver a usar para el propósito de determinar si la partición de MB actual (8x8) se predice a partir de una imagen normal o de una imagen VSP.

[0229] La predicción de vector de movimiento es un proceso de determinación de vectores de movimiento para una unidad de vídeo actual, tal como un MB o una partición de MB. En general, cuando un codificador de vídeo, tal como el codificador de vídeo 20 o el decodificador de vídeo 30, determina un vector de movimiento de la unidad de vídeo actual, el codificador de vídeo puede determinar un valor de profundidad representativo en un componente de vista de profundidad. En algunos ejemplos, el valor de profundidad representativo puede ser un valor de profundidad promedio de los valores de profundidad en un bloque de los valores de profundidad situados junto con los bloques de muestra de la unidad de vídeo actual. El decodificador de vídeo puede determinar entonces, basándose al menos en parte en el valor de profundidad representativo, un vector de disparidad para la unidad de vídeo actual. Un punto de origen del vector de disparidad puede ser una ubicación, dentro de un componente de vista de textura de referencia de la misma unidad de acceso que la unidad de vídeo actual, que está situada con una situación dentro de un bloque de muestra de la unidad de vídeo actual. Un bloque de muestra de una unidad de vídeo de referencia del componente de vista de textura de referencia puede incluir el punto terminal del vector de disparidad. Si la unidad de vídeo de referencia se predice a partir de una vista de textura de otra unidad de acceso, el vector de movimiento de la unidad de vídeo actual puede ser igual al vector de movimiento de la unidad de vídeo de referencia. En otros casos, el vector de movimiento de la unidad de vídeo actual puede ser igual al vector de disparidad.

[0230] En la predicción de parámetros de movimiento basados en la profundidad (DBMP), la información de movimiento, tal como los vectores de movimiento y los índices de referencia, para cada píxel de una unidad de vídeo codificada (por ejemplo,

[0231] un MB, una partición de MB, etc.) se pueden deducir directamente con el uso de mapas de disparidad ya codificados de unidades de vídeo codificadas en la vista cercana en la misma instancia temporal (es decir, la unidad de acceso). En otras palabras, un codificador de vídeo puede determinar vectores de movimiento e índices de referencia para cada píxel basándose en los mapas de disparidad. Este procedimiento puede repetirse independientemente para cada píxel de una unidad de vídeo codificada. En consecuencia, en la DBMP, el vector de movimiento y los índices de referencia no se transmiten en el flujo de bits, sino que se obtienen de la vista de referencia en el decodificador de vídeo 30.

[0232] En algunos ejemplos, la VSP se puede unificar con la predicción del vector de movimiento. Por ejemplo, la VSP se puede simplificar a una deformación hacia atrás sin relleno de orificios. La deformación hacia atrás es la inversa de la deformación. La deformación es un proceso en el que las muestras en los componentes de la vista de referencia se asignan, basándose en la disparidad, a las localizaciones en un componente de vista objetivo. El llenado de orificios

es un proceso de llenado de píxeles que representan partes desocluidas de objetos de fondo en el componente de vista objetivo. Además, cuando la VSP se unifica con la predicción del vector de movimiento, se puede aplicar un valor de disparidad o un valor de profundidad a todo el bloque 4x4. De esta forma, es posible que la predicción de síntesis de visión se pueda simplificar a la compensación de movimiento tradicional, en la que el vector de movimiento se derive de la profundidad o disparidad y esté asociado con cada bloque 4x4 de un MB o de una partición de MB. En este caso, el modo de VSP se rediseña al llamado modo de VSP unificado.

[0233] La sintaxis descrita en las Tablas 3, 4 y 5 anteriores puede reutilizarse para soportar la unificación de la VSP y la predicción de vector de movimiento. Sin embargo, pueden producirse varios problemas cuando la sintaxis descrita en las tablas 3, 4 y 5 se reutilice para soportar la unificación de la predicción de síntesis de visión y la predicción del vector de movimiento. Por ejemplo, en la VSP normal como se describe en las Tablas 3, 4 y 5 anteriores, el vector de movimiento correspondiente a cada bloque es cero, mientras que los vectores de disparidad se pueden usar para derivar los vectores de movimiento para el bloque usando la VSP. Por tanto, el vector de movimiento no siempre debe ser igual a cero. En otro ejemplo, en la VSP normal, no hay predicción de los vectores de movimiento derivados para los MB vecinos codificados con los modos VSP porque los vectores de movimiento son cero. Sin embargo, en el modo de VSP unificado, los vectores de movimiento derivados para los bloques que usan la VSP no siempre son cero.

[0234] De acuerdo con una o más técnicas de esta divulgación, la imagen VSP no se añade a una lista de imágenes de referencia. Por lo tanto, el codificador de vídeo 20 puede no necesitar señalar información de movimiento (por ejemplo, índices de referencia, vectores de movimiento, etc.) para un bloque codificado usando una imagen VSP. Además, dado que la síntesis de visión puede realizarse mediante el modo de VSP unificado, puede ser innecesario que un codificador de vídeo genere la imagen VSP.

[0235] Por ejemplo, cuando la VSP se unifica a la predicción de vector de movimiento basado en la profundidad o de vector de movimiento basado en la disparidad, la señalización del modo de VSP unificado se introduce en el nivel de Mb o de partición de MB para indicar si un MB o una partición de MB se predice a partir de una imagen VSP, usando el mismo diseño de sintaxis que se describe en las tramas 3, 4 y 5 anteriores. Además, cuando el VSP se unifica con la predicción de vector de movimiento basado en la profundidad, cada uno de los bloques que pertenecen al modo de VSP unificado tiene sus vectores de movimiento derivados por el mapa de profundidad o vectores de movimiento de disparidad de otros bloques vecinos espaciales/temporales.

[0236] En algunos ejemplos, un modo de VSP unificado se identifica por uno de un sub_mb_vsp_flag, un mb_part_vsp_flag o de un vsp_mb_flag que es igual a 1. En otros ejemplos, cuando no se señala ninguno de sub_mb_vsp_flag, mb_part_vsp_flag o vsp_mb_flag, un modo de VSP unificado puede identificarse mediante un índice de referencia que se señale a un valor específico. En estos ejemplos, la codificación por entropía (por ejemplo, la codificación por entropía o la decodificación por entropía) del sub_mb_vsp_flag, del mb_part_vsp_flag y del vsp_mb_flag puede ser la misma que se describe en otra parte en esta divulgación.

[0237] Un codificador de vídeo genera una imagen VSP basada en una imagen de referencia. Una cabecera de fragmento puede indicar una vista que contenga la imagen de referencia a partir de la cual se genera la imagen VSP. La memoria descriptiva 3D-AVC ya indica, en una cabecera de fragmento, la vista que se usará para la síntesis de visión. Esta divulgación puede usar "refViewPicVSP" para indicar la imagen decodificada de la vista que se indica en la cabecera de fragmento y que está en la misma unidad de acceso que la imagen de vista de referencia para la VSP. Puede que el codificador de vídeo ya haya añadido refViewPicVSP a la lista de imágenes de referencia porque una vista usada para la síntesis de visión es típicamente una vista que se usa para la predicción inter-vistas de la manera de MVC normal. El codificador de vídeo puede determinar un índice de referencia (correspondiente a RefPicList0) de la siguiente manera. Para cada *i* de 0 a num_ref_idx_l0_active_minus1, inclusive, si RefPicList0 [*i*] es igual a refViewPicVSP, el codificador de vídeo puede establecer el índice de referencia en *i.* num_ref_idx_l0_active_minus1 puede indicar el número de imágenes de referencia activas en RefPicList0, menos 1. Si *i* es igual a num_ref_idx_l0_active_minus1 más 1, el codificador de vídeo puede determinar que el índice de referencia es igual a *i* y refViewPicVSP se añade a RefPicList0 como la última entrada.

[0238] De forma alternativa, cuando los indicadores (por ejemplo, vsp_mb_flag, mb_part_vsp_flag, etc.) no se usan para indicar el modo de VSP unificado y se señala explícitamente un índice de referencia (ref_idx_lx), además, el índice de referencia (ref_idx_lx) puede reiniciarse como sigue. Para cada *i* de 0 a num_ref_idx_lX_active_minus1, inclusive, si RefPicListX [*i*] es igual a RefViewPicVSP, ref_idx_lx se establece en *i.*

[0239] Este proceso de derivación para el índice de referencia se puede usar para reducir el tamaño de las listas de imágenes de referencia en el decodificador de vídeo 30. Prácticamente, no hay ninguna imagen correspondiente al índice de referencia ref_idx_lx que se añada a la lista de imágenes de referencia. En su lugar, ref_idx_lx puede usarse puramente para señalar un modo de VSP unificado y el bloque que se codifica con el modo de VSP unificado se compensa con el movimiento desde la RefPicList[ref_idx_lx], después de que se reinicie ref_idx_lx.

[0240] En ejemplos donde una unidad de vídeo actual (por ejemplo, un MB, una partición de MB o una partición de subMB) se codifique con el modo de síntesis de visión, la derivación de los vectores de movimiento puede incluir derivación, para cada bloque 4x4 de la unidad de vídeo actual, de un vector de movimiento. Los vectores de

movimiento pueden referirse todos a refViewPicVSP y típicamente solo se predicen para RefPicList0, lo que significa que la unidad de vídeo actual solo se predice unidireccionalmente.

[0241] Puede haber múltiples maneras de derivar un vector de movimiento para una unidad de vídeo actual de un componente de vista de textura actual. Por ejemplo, un codificador de vídeo puede derivar el vector de movimiento del correspondiente bloque de vista de profundidad decodificado. En este ejemplo, el codificador de vídeo puede obtener un valor de profundidad representativo. Cuando la imagen de profundidad equivale a ¼ de la resolución de la imagen de textura actual (es decir, un bloque 4x4 en el componente de vista de textura actual corresponde a una región 2x2 en el componente de vista de profundidad), el codificador de vídeo puede procesar los cuatro valores en la región 2x2 de diversas maneras. Por ejemplo, el valor máximo de los cuatro valores se puede devolver como un valor de profundidad representativo. En otro ejemplo, el valor promedio de los cuatro valores se puede devolver como el valor de profundidad representativo. En otro ejemplo, el valor mediano de los cuatro valores se devuelve como el valor de profundidad representativo. Cuando la imagen de profundidad no es ¼ de la resolución de la imagen de textura actual (es decir, un bloque 4x4 en el componente de vista de textura actual corresponde a una región NxM en el componente de vista de profundidad), el codificador de vídeo puede determinar el valor de profundidad representativo de diversas maneras. Por ejemplo, el decodificador de vídeo puede determinar que el valor de profundidad representativo sea el promedio de todos los píxeles de profundidad de la región NxM correspondiente. En otro ejemplo, el decodificador de vídeo puede determinar que el valor de profundidad representativo sea el promedio de los cuatro píxeles de profundidad de esquina de la correspondiente región NxM. En otro ejemplo, el decodificador de vídeo puede determinar que el valor de profundidad representativo sea el máximo de los cuatro píxeles de profundidad de esquina de la correspondiente región NxM. En otro ejemplo, el decodificador de vídeo puede determinar que el valor de profundidad representativo sea el valor medio de los cuatro valores de profundidad de esquina de la correspondiente región NxM. Después de determinar el valor de profundidad representativo, el decodificador de vídeo puede convertir el valor de profundidad representativo en un vector de disparidad y establecer el vector de movimiento en el vector de disparidad. El componente del eje y del vector de disparidad (y el vector de movimiento) puede ser igual a 0.

[0242] Además, puede haber múltiples formas de derivar el vector de movimiento del mapa de profundidad de una vista de referencia cuando la vista de profundidad correspondiente (es decir, la vista con la muestra, view_id) no se haya decodificado. Además, puede haber múltiples formas de derivar el vector de movimiento del bloque vecino espacial/temporal del bloque actual. De forma alternativa, el componente vertical del vector de movimiento derivado puede reiniciarse en 0.

[0243] En algunos ejemplos descritos anteriormente, se puede considerar que dos bloques vecinos codificados con o sin modo de VSP se predicen entre dos imágenes de referencia. Por tanto, la fuerza límite del límite entre los dos bloques vecinos puede ser igual al caso cuando dos bloques vecinos se someten ambos a la interpredicción, pero a partir de dos imágenes de referencia diferentes. Este diseño del filtro de desbloqueo se puede usar para un filtro de desbloqueo cuando se use un modo de VSP unificado en uno de los bloques vecinos.

[0244] Por ejemplo, cuando los dos bloques vecinos se predigan por los modos VSP, un módulo de filtro (tal como el módulo de filtro 113 o 160) puede comparar los vectores de movimiento derivados, similar al caso en el filtro de desbloqueo cuando se predigan ambos bloques a partir de la misma imagen de referencia, y por tanto se comparan los vectores de movimiento. Por otro lado, cuando un bloque se codifica usando el modo de VSP unificado y el otro bloque se codifica con predicción inter-vistas y refViewPicVSP es igual a la imagen de referencia de predicción inter-vistas, se considera que los dos bloques tienen la misma imagen de referencia. Además, cuando un bloque se codifique usando el modo de VSP unificado y el otro bloque se codifique con una imagen de referencia temporal, el bloque codificado con el modo de VSP unificado se puede considerar como un bloque intracodificado.

[0245] Como se describió anteriormente, las técnicas de esta divulgación pueden proporcionar un mecanismo de VSP eficiente para 3DV basada en HEVC. Estas técnicas pueden implementarse de diversas formas. De acuerdo con una primera implementación para 3DV basada en HEVC, un conjunto de parámetros de secuencia puede incluir indicadores para soportar la VSP en 3DV basada en HEVC. La Tabla 28 siguiente muestra una parte de una sintaxis de ejemplo para un conjunto de parámetros de secuencia en 3DV basada en HEVC.

Tabla 28: Sintaxis de RBSP del conjunto de parámetros de secuencia

seq_parameter_set_rbsp() {	**Descriptor**
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
…	
if(sps_extension_flag) {	
…	

seq_parameter_set_rbsp() {	**Descriptor**
seq_vsp_flag	u(1)
if(seq_vsp_flag)	
vsp_in_cu_level_only_flag	u(1)
…	
}	
rbsp_trailing_bits()	
}	

[0246] En el ejemplo de la Tabla 28, el SPS incluye un elemento de sintaxis seq_vsp_flag. Si el elemento de sintaxis seq_vsp_flag tiene un valor particular (por ejemplo, 1), el SPS incluye un elemento de sintaxis vsp_in_cu_level_only_flag. El elemento de sintaxis seq_vsp_flag indica si el modo de VSP está habilitado para todas las imágenes que hacen referencia al SPS. Por ejemplo, seq_vsp_flag igual a 1 puede indicar que el modo de VSP está habilitado para todas las imágenes que hagan referencia al SPS. En este ejemplo, seq_vsp_flag igual a 0 puede indicar que el modo de VSP no está habilitado. Además, en este ejemplo, cuando seq_vsp_flag no está presente, el decodificador de vídeo 30 puede deducir que seq_vsp_flag es igual a 0.

[0247] El elemento de sintaxis de vsp_in_cu_level_only_flag puede indicar si el modo de VSP se aplica solo al nivel de CU. Por ejemplo, vsp_in_cu_level_only_flag igual a 1 puede indicar que el modo de VSP sólo se aplica al nivel de CU. Por tanto, si vsp_in_cu_level_only_flag es igual a 1, se puede predecir una PU a partir de la imagen VSP, pero otra PU puede predecirse por otros modos. En este ejemplo, vsp_in_cu_level_only_flag igual a 0 puede indicar que el modo de VSP se aplica al nivel de PU. Además, en este ejemplo, cuando vsp_in_cu_level_only_flag no está presente, el decodificador de vídeo 30 puede deducir que vsp_in_cu_level_only_flag es igual a 0. En otros ejemplos, vsp_in_cu_level_only_flag no está presente y el modo de VSP siempre se puede aplicar al nivel de PU.

[0248] En la primera implementación de ejemplo para 3DV basada en HEVC, las cabeceras de fragmento pueden incluir indicadores que soporten la VSP en 3DV basada en HEVC. La Tabla 29 siguiente muestra una sintaxis de ejemplo para una cabecera de fragmento en 3DV basada en HEVC.

TABLA 29: Sintaxis de cabecera de fragmento

slice_header() {	**Descriptor**
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag) {	
pic_parameter_set_id	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(**separate_colour_plane_flag** == 1)	
colour_plane_id	u(2)
if(IdrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_**flag**	u(1)
} else {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
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)

slice_header() {	**Descriptor**
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lt[i]	ue(v)
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt_minus1[i]	ue(v)
used_**by_curr_pic_lt_flag**[i]	u(1)
}	
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_**interleaving_flag**	u(1)
slice_sample_adaptive_offset_flag	u(1)
if(slice_sao_interleaving_flag && slice_sample_adaptive_offset_flag) {	
sao_cb_**enable_flag**	u(1)
sao_cr_enable_flag	u(1)
}	
}	
if(scaling_list_enable_flag || deblocking_filter_in_aps_enabled_flag || (sample_adaptive_offset_enabled_flag && !slice_sao_interleaving_flag) || adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
if(slice_type = = P || slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
if(lists_modification_present_flag) {	
ref_pic_list_modification()	
ref_pic_list_combination()	
}	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
}	
if(cabac_init_present_flag && slice_type != I)	
cabac_init_flag	u(1)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
if(deblocking_filter_in_aps_enabled_flag)	

slice_header() {	**Descriptor**
inherit_dbl_params_from_aps_flag	u(1)
if(!inherit_dbl_params_from_aps_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
}	
if(seq_vsp_flag && slice_type !=I)	
slice_vsp_enable_flag	u(1)
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if(slice_type !=I && ((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num ref idx 11 active minusl > 0))	
collocated_ref_idx	ue(v)
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_idc = = 1 && slice_type = = B))	
pred_weight_table()	
}	
if(slice_type = = P || slice_type = = B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_sfice_flag)	
alf_param()	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_sfice_flag)	
alf_cu_control_param()	
}	
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(1)
if(tiles_or_entropy_coding_sync_idc > 0) {	
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)
}	
}	
}	

[0249] En la sintaxis de ejemplo de la Tabla 29, la cabecera de fragmento de un fragmento incluye un elemento de sintaxis slice_vsp_enable_flag si el elemento de sintaxis seq_vsp_flag del SPS aplicable indica que el modo de VSP está habilitado y el fragmento no es un fragmento I. El elemento de sintaxis slice_vsp_enable_flag de la cabecera de fragmento puede indicar si la VSP está habilitada para el fragmento. Por ejemplo, slice_vsp_enable_flag igual a 1 puede indicar que la VSP está habilitada para el fragmento actual. En este ejemplo, slice_vsp_enable_flag igual a 0

puede indicar que la VSP no está habilitada para el fragmento actual. Además, en este ejemplo, cuando slice_vsp_enable_flag no está presente, el decodificador de vídeo 30 puede deducir que slice_vsp_enable_flag es igual a 0.

[0250] En la primera implementación para 3DV basada en HEVC, las CU pueden incluir indicadores que soporten la VSP en 3DV basada en HEVC. La Tabla 30 siguiente muestra una sintaxis de ejemplo para una CU en 3DV basada en HEVC.

TABLA 30 - Sintaxis de unidad de codificación

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
CurrCbAddrTS = MinCbAddrZS[x0 >> Log2MinCbSize][y0 >> Log2MinCbSize]	
if(slice_type != I) {	
if (slice_vsp_enable_flag)	
vsp_cu_ftag[x0**][**y0**]**	ae(v)
if(!vsp_cu_flag[x0][y0])	
skip_flag[x0**][**y0**]**	ae(v)
}	
if(!vsp_cu_flag[x0][y0] && skip_flag[x0][y0])	
prediction_unit(x0, y0, log2CbSize)	
else if(!vsp_cu_flag[x0][y0] && (slice_type != I || log2CbSize = = Log2MinCbSize)) {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(PredMode != MODE_INTRA || log2CbSize = = Log2MinCbSize)	
part_mode	ae(v)
x1 = x0 + ((1 << log2CbSize) >> 1)	
y1 = y0 + ((1 << log2CbSize) >> 1)	
x2 = x1 - ((1 << log2CbSize) >> 2)	
y2 = y1 - ((1 << log2CbSize) >> 2)	
x3 = x1 + ((1 << log2CbSize) >> 2)	
y3 = y1 + ((1 << log2CbSize) >> 2)	
if(PartMode = = PART_2Nx2N)	
prediction_unit(x0, y0, log2CbSize)	
else if(PartMode = = PART_2NxN) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x0, y1, log2CbSize)	
} else if(PartMode = = PART Nx2N) {	
prediction unit(x0, y0, log2CbSize)	
prediction_unit(x1, y0, log2CbSize)	
} else if(PartMode = = PART 2NxnU) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x0, y2, log2CbSize)	
} else if(PartMode = = PART 2NxnD) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x0, y3, log2CbSize)	
} else if(PartMode = = PART_nLx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x2, y0, log2CbSize)	

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
} else if(PartMode = = PART_nRx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x3, y0, log2CbSize)	
} else {/* PART_NxN */	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x1, y0, log2CbSize)	
prediction_unit(x0, y1, log2CbSize)	
prediction_unit(x1, y1, log2CbSize)	
}	
}	
if(!skip_flag[x0][y0]&&!pcm_flag)	
transform_tree(x0, y0, x0, y0, log2CbSize, log2CbSize, log2CbSize, 0, 0)	
}	

[0251] En el ejemplo de la Tabla 30, la CU incluye un elemento de sintaxis vsp_cu_flag si el elemento de sintaxis slice_vsp_enable_flag de la cabecera de fragmento del fragmento actual (es decir, el fragmento que incluye la CU) indica que la VSP está habilitada para el fragmento actual. El elemento de sintaxis vsp_cu_flag[x0][y0] puede indicar que una CU que incluya un píxel en las coordenadas (x0, y0) de la imagen actual se predice a partir de una imagen VSP. Es decir, los índices de matriz x0, y0 pueden especificar la ubicación (x0, y0) de la muestra de luminancia superior izquierda de la región de la CU actual (es decir, el bloque de codificación considerado) relativa a la muestra de luminancia superior izquierda de la imagen actual. Por ejemplo, cuando la CU actual esté en un fragmento P o B, vsp_cu_flag[x0][y0] igual a 1 puede especificar que los elementos de sintaxis pred_mode_flag y part_mode de la CU actual no están presentes, que la CU actual no incluye ninguna PU y que toda la CU actual se predice a partir de la imagen VSP. En este ejemplo, vsp_cu_flag[x0][y0] igual a 0 puede especificar que toda la CU actual no se predice totalmente desde una imagen VSP. En consecuencia, si vsp_cu_flag[x0][y0] es igual a 0, la CU actual puede incluir una o más PU. Además, en este ejemplo, si vsp_cu_flag[x0][y0] no está presente, el decodificador de vídeo 30 puede deducir que vsp_cu_flag[x0][y0] es igual a 0.

[0252] Además, en la primera implementación de ejemplo para 3DV basada en HEVC, las PU pueden incluir indicadores que soporten la VSP. La Tabla 31 siguiente muestra una sintaxis de ejemplo para una PU en 3DV basada en HEVC.

TABLA 31 - Sintaxis de unidad de predicción

prediction_unit(x0, y0, log2CbSize){	**Descriptor**
if (!vsp_cu_flag)	
if(PartMode != PART_2Nx2N && !vsp_in_cu_level_only_flag)	
vsp_pu_flag[x0][y0]	ae(v)
}	
if (!vsp_pu_flag[x0][y0]){	
if(skip_flag[x0][y0]){	
if(MaxNumMergeCand >1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else if(PredMode == MODE_INTRA){	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIPCMCUSize && log2CbSize <= Log2MaxIPCMCUSize)	
pcm_flag	ae(v)
if(pcm_flag){	
num_subsequent_pcm	tu(3)
NumPCMBlock = num_subsequent_pcm + 1	

prediction_unit(x0, y0, log2CbSize){	**Descriptor**
while(!byte_aligned())	
pcm_alignment_zero_bit	u(v)
pcm_sample(x0, y0, log2CbSize)	
}else{	
prev_intra_luma_pred_flag[x0][y0]	ae(v)
if(prev_intra_luma_pred_flag[x0][y0])	
mpm_idx[x0][y0]	ae(v)
Else	
rem_intra_luma_pred_mode[x0][y0]	ae(v)
intra_chroma_pred_mode[x0][y0]	ae(v)
SignalledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_ _mode[x0][y0] == 3: intra_chroma_pred_mode[x0][y0] == 2)	
}	
} else {/* MODE_INTER */	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else {	
if(slice_type == B)	
inter_pred_flag[x0][y0]	ae(v)
if(inter_pred_flag[x0][y0] == Pred_LC){	
if(num_ref_idx_lc_active_minus1 > 0)	
ref_idx_lc[x0][y0]	ae(v)
mvd_coding(mvd_lc[x0][y0][0], mvd_lc[x0][y0][1])	
mvp_lc_flag[x0][y0]	ae(v)
} else {/* Pred_L0 or Pred BI */	
if(num_ref_idx_10_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(mvd_l0[x0][y0][0], mvd_l0[x0][y0][1])	
mvp_l0_flag[x0][y0]	ae(v)
}	
if(inter_pred_flag[x0][y0] == Pred_BI) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1 zero flag) {	
mvd_l1[x0][y0][0] = 0	
mvd_l1[x0][y0][1] = 0	
} else	
mvd_coding(mvd_l1[x0][y0][0], mvd_l1[x0][y0][1])	
mvp_l1_flag[x0][y0]	ae(v)
}	
}	

prediction_unit(x0, y0, log2CbSize){	**Descriptor**
}	
}	
}	

[0253] En la sintaxis de la Tabla 31, una PU puede incluir un elemento de sintaxis vsp_pu_flag[x0][y0] si el vsp_cu_flag para la CU indica que toda la CU no se predice a partir de una VSP, si la CU no está particionada de acuerdo con el modo de partición 2Nx2N y si el vsp_in_cu_level_only_flag del SPS aplicable no indica que el modo de VSP sólo se aplica al nivel de CU. El elemento de sintaxis vsp_pu_flag[x0][y0] puede indicar si la PU se codifica con la predicción VSP. En otras palabras, vsp_pu_flag[x0][y0] puede indicar si la PU se predice a partir de una imagen VSP. Por ejemplo, vsp_pu_flag[x0][y0] igual a 1 puede especificar que la PU actual se codifica con la predicción VSP y que otros elementos relacionados con la PU no están presentes en la PU. En este ejemplo, vsp_pu_flag [x0] [y0] igual a 0 especifica que la PU actual no se codifica con la predicción VSP. Cuando vsp_pu_flag[x0][y0] no está presente, el decodificador de vídeo 30 puede deducir que vsp_pu_flag[x0][y0] es lo mismo que vsp_cu_flag[x0][y0] de la CU que contiene la PU. En otro ejemplo, el flujo de bits está restringido de tal manera que no todas las PU dentro de una CU pueden tener vsp_pu_flag igual a 1 si tanto vsp_cu_flag como vsp_in_cu_level_only_flag son iguales a 0.

[0254] En el ejemplo de la Tabla 31 anterior, vsp_cu_flag puede ser el valor de vsp_cu_flag de la CU que contenga esta PU. De forma alternativa, en la Tabla 31, la condición "si (vsp_cu_flag)" se puede cambiar para especificar explícitamente vsp_cu_flag basándose en las coordenadas y en el tamaño de la CU:

si (vsp_cu_flag[x0>>log2CbSize<<log2CbSize][y0>>log2CbSize<<log2CbSize])

[0255] Además, en la primera implementación de ejemplo para 3DV basada en HEVC, un codificador de vídeo realiza un proceso para generar un bloque predictivo para una CU o una PU cuando la CU o la PU use el modo de VSP. En algunos ejemplos, la CU o la PU pueden usar el modo de VSP si el elemento de sintaxis vsp_cu_flag o el elemento de sintaxis vsp_pu_flag aplicable a la PU es igual a 1. En este proceso, la CU o PU actual puede tener un píxel superior izquierdo con las coordenadas (x, y) en el componente de luminancia y (x ', y') en los componentes de crominancia. Además, en este proceso, se supone que el tamaño de la CU o de la PU actual es NxM. Partiendo de las coordenadas (x', y') con el tamaño de N/2xM/2 en el caso del formato de vídeo 4:2:0, NxM en el caso del formato de vídeo 4:4:4 o NxM/2 en el caso del formato de vídeo 4:2:2, la unidad de compensación de movimiento 164 puede establecer los valores de píxeles en cada bloque de un componente de crominancia de la imagen VSP en los píxeles de la CU o de la PU actual, comenzando con las mismas coordenadas del componentes de crominancia (x', y'). Por ejemplo, la unidad de compensación de movimiento 164 puede generar un bloque Cb o Cr predictivo para la CU o la PU actual de tal manera que cada muestra Cb o Cb coincida con una muestra Cb o Cr en una ubicación correspondiente en la imagen VSP. En este ejemplo, la unidad de compensación de movimiento 164 puede comenzar a generar el bloque Cb o Cr predictivo a partir de las coordenadas (x', y') de un bloque Cb o Cr de la imagen VSP. Además, en este ejemplo, el bloque Cb o Cr predictivo puede ser de tamaño N/2xM/2, NxM o nxM/2 si los bloques de crominancia de la imagen VSP se muestrean de forma insuficiente de acuerdo con el formato de vídeo 4:2:0, el formato de vídeo 4:4:4 o formato de vídeo 4:2:2, respectivamente.

[0256] Además, en la primera implementación de ejemplo para 3DV basada en HEVC, un codificador de vídeo puede mantener un contexto para vsp_cu_flag. El contexto de vsp_cu_flag se puede usar para proporcionar el contexto para codificar por entropía tanto vsp_pu_flag como vsp_cu_flag. El codificador de vídeo puede actualizar el contexto para el indicador vsp_cu cuando el codificador de vídeo codifique por entropía tanto vsp_pu_flags como vsp_cu_flags. Cuando vsp_cu_flag es igual a 1, se deduce que cada PU de CU tiene VspFlag igual a 1. Además, cuando vsp_cu_flag es igual a 1, cada bloque 4x4 en la CU tiene VspFlag igual a 1. Cuando vsp_cu_flag es igual a 0, las PU en la CU pueden tener diferentes valores de vsp_pu_flag. Si una PU tiene vsp_pu_flag igual a 1, entonces VspFlags para todos los bloques 4x4 en la PU se establecen en 1.

[0257] En la primera implementación de ejemplo para 3DV basada en HEVC, un codificador de vídeo puede inicializar un contexto para el VspFlag. La probabilidad inicial para el contexto del VspFlag de los fragmentos P puede ser diferente de la de los fragmentos B. VspFlag se puede aplicar a los fragmentos P y B. Además, en la primera implementación de ejemplo para 3DV basada en HEVC, cuando el codificador por entropía esté codificando por entropía el VspFlag, el codificador por entropía puede determinar el ctxIdxInc para un bin asociado con el VspFlag. En algunos ejemplos, el ctxIdxInc para el bin puede ser la suma de condTermFlagA y condTermFlagB. Es decir, ctxIdxInc se puede derivar como ctxIdxInc = condTermFlagA + condTermFlagB. En otro ejemplo, el ctxIdxInc para el bin puede ser igual a condTermFlagA. Es decir, ctxIdxInc se puede derivar como ctxIdxInc = condTermFlagA. En otro ejemplo, el ctxIdxIdc para el bin es igual a condTermFlagB. Es decir, ctxIdxInc se puede derivar como ctxIdxInc = condTermFlagB. En otro ejemplo, ctxIdxInc se fija para ser igual a 0.

[0258] En los ejemplos anteriores, el codificador por entropía puede determinar condTermFlagA y condTermFlagB basándose al menos en parte en si los datos asociados con bloques particulares que estén al lado de una CU o de una PU actual están disponibles para su uso en el proceso de codificación de entropía para los elementos de sintaxis

asociados con la CU o o con la PU actual. Por ejemplo, uiPAddrA puede indicar el bloque 4x4 vecino a la izquierda de la CU o de la PU actual y uiPAddrB puede indicar el bloque 4x4 vecino encima de la CU o de la PU actual. En algunos ejemplos, si uiPAddrN (donde N es A o B) no está disponible o VspFlag para el bloque uiPAddrN es igual a 0, condTermFlagN es igual a 0. De lo contrario, en este ejemplo, si uiPAddrN está disponible y VspFlag para el bloque uiPAddrN es igual a 1), condTermFlagN es igual a 1.

[0259] En el primer ejemplo de implementación para 3DV basada en HEVC, una unidad de filtro (tal como la unidad de filtro 114 del codificador de vídeo 20 o la unidad de filtro 160 del decodificador de vídeo 30) puede realizar un proceso de filtro de desbloqueo. En este proceso, la unidad de filtro calcula una resistencia de límite (bS) para un límite entre dos píxeles vecinos p0 y q0. En este ejemplo, la unidad de filtro puede determinar una resistencia límite relacionada con una región VSP. La región VSP puede ser una región de PU o de CU adyacentes que se codifiquen con el modo de VSP.

[0260] Para determinar la resistencia límite relacionada con la región VSP, la unidad de filtro puede determinar si el límite de p0 y q0 cruza el límite de una región VSP. Si el límite de p0 y q0 no cruza el límite de una región de VSP, la unidad de filtro puede establecer la resistencia límite en la misma resistencia de límite que para el caso cuando el límite de p0 y q0 se encuentre dentro de una región sometida a interpredicción. Una región sometida a interpredicción es una región de PU o de CU adyacentes que se codifican todas con los mismos pares de imagen/imágenes de referencia y vectores de movimiento iguales o similares (es decir, el mismo después de redondearse a una cierta precisión).

[0261] Por otro lado, si el límite de p0 y q0 cruza el límite entre una región VSP y una región sometida a interpredicción, la unidad de filtro puede establecer la fuerza límite a la misma resistencia de límite que para el caso cuando el límite cruce dos regiones sometidas a interpredicción. Cuando p0 o q0 pertenezca a una CU intracodificada, la unidad de filtro puede establecer la intensidad límite en la misma intensidad límite que en el caso en que p0 pertenezca a una CU sometida a interpredicción y que q0 pertenezca a una región sometida a interpredicción. Cuando p0 pertenece a una CU intracodificada, q0 pertenece a una región de VSP. Del mismo modo, cuando q0 pertenece a una CU intracodificada, p0 pertenece a una región de VSP. Una región sometida a interpredicción puede ser una CU codificada con un cierto modo de intrapredicción.

[0262] Una segunda implementación de ejemplo para 3DV basada en HEVC puede ser similar a la primera implementación de ejemplo para 3DV basada en HEVC. Sin embargo, en el segundo ejemplo de implementación para 3DV basada en HEVC, la VSP se aplica solo al nivel de CU, no al nivel de PU. En este segundo ejemplo de implementación para 3DV basada en HEVC, la sintaxis y la semántica de las PU no se modifican en relación con el perfil principal descrito en el Borrador de Trabajo 6 de la HEVC. Además, a diferencia de la primera implementación de ejemplo para 3DV basada en HEVC, la codificación por entropía de vsp_cu_flag puede basarse únicamente en vsp_cu_flag porque en este segundo ejemplo de implementación para 3DV basada en HEVC no existe vsp_pu_flag. En este segundo ejemplo de implementación para 3DV basada en HEVC, el codificador de vídeo 20 no señala vsp_in_pu_level_flag y el decodificador de vídeo 30 siempre puede deducir (es decir, determinar automáticamente) vsp_in_pu_level_flag para ser igual a 0.

[0263] Una tercera implementación de ejemplo para 3DV basada en HEVC puede ser similar a la primera implementación de ejemplo para 3DV basada en HEVC. Sin embargo, la tercera implementación de ejemplo para 3DV basada en HEVC puede tener una estructura de sintaxis diferente para la CU que la primera implementación de ejemplo para 3DV basada en HEVC. En la tercera implementación de ejemplo para 3DV basada en HEVC, la estructura de sintaxis para una CU se describe en la Tabla 32 siguiente.

TABLA 32 - Sintaxis de unidad de codificación

coding_unit(x0, y0, log2CbSize){	**Descriptor**
CurrCbAddrTS = MinCbAddrZS[x0 >> Log2MinCbSize][y0 >> Log2MinCbSize]	
if(slice_type != I) {	
skip_flag[x0][y0]	ae(v)
if (slice_vsp_enable_flag)	
vsp_cu_ski p_flag [x0][y0]	ae(v)
}	
if(skip_flag[x0][y0] &&!vsp_cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, log2CbSize)	
else if(slice_type !=I || log2CbSize == Log2MinCbSize){	
if (slice_vsp_enable_flag && slice_type != I)	

coding_unit(x0, y0, log2CbSize){	**Descriptor**
vsp_cu_ftag[x0][y0]	ae(v)
if(!vsp_cu_flag[x0][y0]){	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(PredMode != MODE_INTRA || log2CbSize == Log2MinCbSize)	
part_mode	ae(v)
x1=x0+((1<<log2CbSize)>>1)	
y1=y0+((1<<log2CbSize)>>1)	
x2 = x1 -((1<< log2CbSize)>>2)	
y2 = y1 -((1 <<log2CbSize)>>2)	
x3 = x1 +((1<< log2CbSize)>>2)	
y3 = y1 +((1<< log2CbSize)>>2)	
if(PartMode == PART_2Nx2N)	
prediction_unit(x0, y0, log2CbSize)	
else if(PartMode == PART_2NxN) {	
prediction_unit(x0,y0, log2CbSize)	
prediction_unit(x0, y1, log2CbSize)	
} else if(PartMode = = PART_Nx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction unit(x1, y0, log2CbSize)	
} else if(PartMode = =PART_2NxnU) {	
prediction_unit(x0, y0, log2CbSize)	
prediction unit(x0, y2, log2CbSize)	
} else if(PartMode == PART 2NxnD) {	
prediction_unit(x0, y0, log2CbSize)	
prediction unit(x0, y3, log2CbSize)	
} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction unit(x2, y0, log2CbSize)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction unit(x3, y0, log2CbSize)	
} else {/* PART_NxN */	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x1, y0, log2CbSize)	
prediction_unit(x0, y1, log2CbSize)	
prediction_unit(x1, y1, log2CbSize)	
}	
}	
if(pcm_flag)	
transform_tree(x0, y0, x0, y0, log2CbSize, log2CbSize, log2CbSize, 0, 0)	
}	

[0264] La estructura de sintaxis de ejemplo de la Tabla 32 incluye un skip_flag[x0][y0]. skip_flag[x0][y0] igual a 1 especifica que, para la CU actual, al decodificar un fragmento P o B, no se pueden decodificar más elementos de sintaxis excepto los índices de predicción del vector de movimiento después de skip_flag[x0][y0]. skip_flag[x0][y0] igual a 0 especifica que no se omite la CU. Los índices de matriz x0, y0 especifican la ubicación (x0, y0) de la muestra de luminancia superior izquierda del bloque de codificación considerado en relación con la muestra de luminancia superior izquierda de la imagen. Cuando skip_flag[x0][y0] no está presente, el decodificador de vídeo 30 puede deducir que skip_lag[x0][y0] es igual a 0. Una CU con skip_flag[x0][y0] igual a 1 puede ser una CU de omisión convencional en HEVC o una CU codificada con la VSP y no tiene ningún residuo señalado, por lo que la estructura de sintaxis de transform_tree no está presente en la CU.

[0265] De acuerdo con una o más técnicas de esta divulgación, la CU puede incluir un elemento de sintaxis vsp_cu_skip_flag[x0][y0] si slice_vsp_enable_flag es igual a 1. El elemento de sintaxis vsp_cu_skip_flag[x0][y0] indica si el modo de VSP predice la CU actual. Por ejemplo, vsp_cu_skip_flag[x0][y0] igual a 1 especifica que la CU actual se codifica usando el modo de omisión y se predice a partir de la imagen VSP. En este ejemplo, vsp_cu_skip_flag[x0][y0] igual a 0 especifica que la CU actual se predice con el modo de omisión de la HEVC convencional. Cuando una CU está en un fragmento P o B y la CU se codifica en el modo de omisión de HEVC convencional, no se decodifican más elementos de sintaxis excepto los índices de predicción vectorial de movimiento después de skip_flag[x0][y0]. Por lo tanto, de acuerdo con una o más técnicas de esta divulgación, la CU puede incluir una PU si skip_flag[x0][y0] es igual a 1 y vsp_cu_skip_flag[x0][y0] no es igual a 1.

[0266] Además, si skip_flag[x0][y0] no es igual a 0 o vsp_cu_skip_flag[x0][y0] es igual a 1, y el fragmento actual no es un fragmento I y el slice_vsp_enable_flag es igual a 1, el CU puede, de acuerdo con una o más técnicas de esta divulgación, incluir un elemento de sintaxis vsp_cu_flag[x0][y0]. vsp_cu_flag[x0][y0] puede indicar si, al decodificar un fragmento P o B, pred_mode_flag, part_mode o el elemento de sintaxis en la tabla de sintaxis de la unidad de predicción están presentes en la CU y toda la CU se predice a partir de la imagen VSP. Por ejemplo, vsp_cu_flag [x0] [y0] igual a 1 puede especificar que, para la CU actual, al decodificar un fragmento P o B, y pred_mode_flag, part_mode o elementos de sintaxis en la tabla de sintaxis de la unidad de predicción no estén presentes y se predice toda la CU de la imagen VSP. En este ejemplo, vsp_cu_flag [x0] [y0] igual a 0 especifica que toda la CU no se predice por completo a partir de una imagen VSP. Los índices de matriz x0, y0 especifican la situación (x0, y0) de la muestra de luminancia superior izquierda del bloque de codificación considerado en relación con la muestra de luminancia superior izquierda de la imagen. Cuando no está presente, se deduce que vsp_cu_flag [x0] [y0] es igual a max(0, vsp_cu_skip_flag[x0][y0]).

[0267] En la tercera implementación de ejemplo para 3DV basada en HEVC, una unidad de codificación por entropía, tal como la unidad de codificación por entropía 118 de la FIG. 2 o la unidad de decodificación por entropía 150 de la FIG. 3, puede mantener un contexto de codificación por entropía para vsp_cu_flag. La unidad de codificación por entropía puede usar el contexto de la codificación por entropía para vsp_cu_flag para la codificación de vsp_pu_flag, vsp_cu_skip_flag y vsp_cu_flag. La unidad de codificación por entropía puede actualizar el contexto de codificación por entropía para el indicador vsp_cu cuando la unidad de codificación por entropía codifique vsp_cu_skip_flags, vsp_cu_flags y vsp_cu_flags. Cuando vsp_cu_flag o vsp_cu_skip_flag sea igual a 1, puede deducirse que cada PU de la CU tiene VspFlag igual a 1. Además, cada bloque 4x4 en la CU tiene VspFlag igual a 1. Además, cuando vsp_cu_flag es 0, las PU en la CU pueden tener diferentes valores de vsp_pu_flag. Si una PU tiene vsp_pu_flag igual a 1, entonces VspFlag para todos los bloques 4x4 en la PU se establece en 1.

[0268] En la tercera implementación de ejemplo para 3DV basada en HEVC, el modelado de contexto para el VspFlag incluye la inicialización de contexto y la selección de contexto. En la inicialización de contexto, el modo de VSP se aplica tanto para los fragmentos P como para los fragmentos B. La probabilidad inicial para el contexto de VspFlag de los fragmentos P puede ser diferente de los fragmentos B.

[0269] En la selección de contexto, uiPAddrA indica el bloque 4x4 vecino a la izquierda de la CU o de la PU actual y uiPAddrB indica el bloque 4x4 vecino encima de la CU o de la PU actual. Además, la unidad de codificación por entropía puede determinar la variable condTermFlagN (con N como A o B). Si uiPAddrA no está disponible o VspFlag para el bloque uiPAddrA es igual a 0, la unidad de codificación por entropía puede establecer condTermFlagN igual a 0. De lo contrario, si uiPAddrN está disponible y VspFlag para el bloque uiPAddrN es igual a 1, la unidad de codificación por entropía puede establecer condTermFlagN igual a 1.

[0270] Además, en el tercer ejemplo de implementación para 3DV basada en HEVC, la unidad de codificación por entropía puede usar un índice de contexto ctxIdx para codificar el VspFlag. La unidad de codificación por entropía puede determinar un incremento al ctxIdx (es decir, el ctxIdxInc) como ctxIdxInc = condTermFlagA + condTermFlagB. En un ejemplo alternativo, la unidad de codificación por entropía puede derivar ctxIdxInc de tal manera que ctxIdxInc = condTermFlagA. En otro ejemplo alternativo, la unidad de codificación por entropía puede derivar ctxIdxInc de tal manera que ctxIdxInc = condTermFlagB. En otro ejemplo alternativo, ctxIdxInc se puede fijar en cero.

[0271] Una cuarta implementación de ejemplo para 3DV basada en HEVC puede ser similar a la primera implementación de ejemplo para 3DV basada en HEVC. Sin embargo, en la cuarta implementación de ejemplo para 3DV basada en HEVC, la sintaxis para las CU puede definirse en la Tabla 33, a continuación.

TABLA 33 - Sintaxis de unidad de codificación

coding_unit(x0, y0, log2CbSize){	**Descriptor**
CurrCbAddrTS = MinCbAddrZS[x0 >> Log2MinCbSize][y0 >> Log2MinCbSize]	
if(slice_type != 1) {	
if (slice_vsp_enable_flag)	
vsp_cu_flag[x0][y0]	ae(v)
skip_flag[x0][y0]	ae(v)
}	
if(!vsp_cu_flag[x0][y0] && skip_flag[x0][y0])	
prediction_unit(x0, y0, log2CbSize)	
else if(!vsp_cu_flag[x0][y0] && (slice_type != I || log2CbSize == Log2MinCbSize)){	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(PredMode != MODE_INTRA || log2CbSize == Log2MinCbSize)	
part_mode	ae(v)
x1 = x0 +((1 <<log2CbSize)>>1)	
y1 = y0 +((1<< log2CbSize)>>1)	
x2 = x1 -((1<< log2CbSize)>>2)	
y2 = y1 -((1 <<log2CbSize)>>2)	
x3 = xl +((1<< log2CbSize)>>2)	
y3 = y1 +((1<< log2CbSize)>>2)	
if(PartMode == PART_2Nx2N)	
prediction_unit(x0, y0, log2CbSize)	
else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x0, y1, log2CbSize)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x1, y0, log2CbSize)	
}else if(PartMode == PART_2NxnU) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x0, y2, log2CbSize)	
} else if(PartMode = = PART_2NxnD) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x0, y3, log2CbSize)	
} else if(PartMode = =PART_nLx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x2, y0, log2CbSize)	
} else if(PartMode = =PART_nRx2N) {	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x3, y0, log2CbSize)	
} else {/* PART_NxN */	
prediction_unit(x0, y0, log2CbSize)	
prediction_unit(x1, y0, log2CbSize)	

coding_unit(x0, y0, log2CbSize){	**Descriptor**
prediction_unit(x0, y1, log2CbSize)	
prediction_unit(x1, y1, log2CbSize)	
}	
}	
if(!skip_flag[x0][y0]&&!pcm_flag)	
transform tree(x0, y0, x0, y0, log2CbSize, log2CbSize, log2CbSize, 0, 0)	
}	

[0272] En la sintaxis de ejemplo de la Tabla 33, la CU incluye vsp_cu_flag[x0][y0]. vsp_cu_flag [x0] [y0] para una CU puede indicar si, al decodificar un fragmento P o B, pred_mode_flag, part_mode o elementos de sintaxis en la tabla de sintaxis de la unidad de predicción no están presentes y toda la CU se predice a partir de la imagen VSP. Por ejemplo, vsp_cu_flag[x0][y0] igual a 1 puede especificar que, para la CU actual, al decodificar un fragmento P o B, y pred_mode_flag, part_mode o elementos de sintaxis en la tabla de sintaxis de PU no estén presentes y que se prediga toda la CU a partir de la imagen VSP. En este ejemplo, vsp_cu_flag[x0][y0] igual a 0 especifica que toda la CU no se predice por completo a partir de una imagen VSP. Los índices de matriz x0, y0 especifican la ubicación (x0, y0) de la muestra de luminancia superior izquierda del bloque de codificación considerado en relación con la muestra de luminancia superior izquierda de la imagen. Cuando vsp_cu_flag [x0] [y0] no está presente, se puede deducir que vsp_cu_flag [x0] [y0] es igual a 0.

[0273] Además, la CU puede incluir un skip_flag [x0] [y0] que indique si la CU actual, al decodificar un fragmento P o B, incluye elementos de sintaxis distintos de los índices de predicción del vector de movimiento (por ejemplo, merge_idx, mvp_lc_flag, mvp_l0_flag, mvp_l1_flag, etc.). Por ejemplo, skip_flag [x0] [y0] igual a 1 especifica que, para la CU actual, al decodificar un fragmento P o B, no se pueden decodificar más elementos de sintaxis excepto los índices de predicción vectorial de movimiento después de skip_flag[x0][y0]. En este ejemplo, skip_flag [x0] [y0] igual a 0 especifica que no se omite la CU. Los índices de matriz x0, y0 especifican la situación (x0, y0) de la muestra de luminancia superior izquierda del bloque de codificación considerado en relación con la muestra de luminancia superior izquierda de la imagen. Cuando skip_flag[x0][y0] no está presente, se puede deducir que skip_flag[x0][y0] es igual a 0. Una CU con skip_flag igual a 1 puede ser una CU de omisión convencional en la HEVC o una CU codificada con la VSP y no tiene señal de residuo, por lo que la estructura de sintaxis del árbol de transformada no está presente. La estructura de sintaxis del árbol de transformada puede incluir información residual para la CU.

[0274] La FIG. 4a es un diagrama de flujo que ilustra una operación 200 de ejemplo de un codificador de vídeo 20 de acuerdo con una o más técnicas de esta divulgación. El diagrama de flujo de la FIG. 4A y los diagramas de flujo de las siguientes figuras se proporcionan como ejemplos. En otros ejemplos, las técnicas de esta divulgación se pueden implementar usando más, menos, o diferentes etapas que las mostradas en el ejemplo de la FIG. 4A y de las siguientes figuras.

[0275] En el ejemplo de la FIG. 4A, el codificador de vídeo 20 puede generar, basándose al menos en parte en un componente de vista de textura previamente codificado de una unidad de acceso actual y en un componente de vista de profundidad de la unidad de acceso actual, una imagen VSP (201). Además, el codificador de vídeo 20 puede indicar, en un flujo de bits que incluya una representación codificada de múltiples vistas de textura y vistas de profundidad, un elemento de sintaxis que indique si una unidad de vídeo actual se predice a partir de una imagen VSP (202). En algunos ejemplos, la unidad de vídeo actual puede ser un MB o una partición de MB de un componente de vista de textura actual de una unidad de acceso actual. En al menos algunos ejemplos en los que la unidad de vídeo actual es un MB, el codificador de vídeo 20 puede indicar, en el flujo de bits, una estructura de sintaxis de la capa de MB que incluya el elemento de sintaxis. En dichos ejemplos, el elemento de sintaxis puede ser vsp_mb_flag. En al menos algunos ejemplos donde la unidad de vídeo actual es una partición de MB, el codificador de vídeo 20 puede indicar, en el flujo de bits, una estructura de sintaxis que incluya el elemento de sintaxis. En dichos ejemplos, la estructura de sintaxis puede ser una estructura de sintaxis de predicción de subMB o una estructura de sintaxis de predicción de subMB y el elemento de sintaxis puede ser mb_part_vsp_flag o sub_mb_vsp_flag.

[0276] Además, el codificador de vídeo 20 puede determinar, basándose al menos en parte en si la unidad de vídeo actual se predice a partir de la imagen VSP, si señalar, en el flujo de bits, la información de movimiento para la unidad de vídeo actual (204). En respuesta a determinar que la unidad de vídeo actual no se predice a partir de una imagen VSP ("NO" de 204), el codificador de vídeo 20 puede señalar, en el flujo de bits, información de movimiento para la unidad de vídeo actual (206). De lo contrario ("SÍ" de 204), el codificador de vídeo 20 puede omitir, del flujo de bits, la información de movimiento para la unidad de vídeo actual (208). En cualquier caso, el codificador de vídeo 20 puede emitir el flujo de bits (210). La información de movimiento para la unidad de vídeo actual puede incluir un índice de referencia y una diferencia de vector de movimiento. El índice de referencia puede indicar una posición dentro de una lista de imágenes de referencia (por ejemplo, RefPicListO o RefPicList1) y la diferencia del vector de movimiento puede

indicar una diferencia entre un predictor de vector de movimiento y un vector de movimiento de la unidad de vídeo actual.

[0277] La FIG. 4B es un diagrama de flujo que ilustra una operación de ejemplo 250 del decodificador de vídeo 30, de acuerdo con una o más técnicas de esta divulgación. En el ejemplo de la FIG. 4B, el decodificador de vídeo 30 puede generar, basándose al menos en parte en un componente de vista de textura previamente codificado de una unidad de acceso actual y en un componente de vista de profundidad de la unidad de acceso actual, una imagen VSP (251). Además, el decodificador de vídeo 30 puede decodificar, a partir de un flujo de bits que incluya una representación codificada de múltiples vistas de textura y múltiples vistas de profundidad, un elemento de sintaxis que indique si una unidad de vídeo actual se predice a partir de una imagen VSP (252). En algunos ejemplos, la unidad de vídeo actual puede ser un MB o una partición de MB de un componente de vista de textura actual de una vista actual de la unidad de acceso actual. En al menos algunos ejemplos donde la unidad de vídeo actual es un MB, el decodificador de vídeo 30 puede decodificar, a partir del flujo de bits, una estructura de sintaxis de capa de MB que incluya el elemento de sintaxis. En algunos de dichos ejemplos, el elemento de sintaxis puede ser vsp_mb_flag. En al menos algunos ejemplos donde la unidad de vídeo actual es una partición de MB, el decodificador de vídeo puede decodificar, a partir del flujo de bits, una estructura de sintaxis que incluya el elemento de sintaxis. En dichos ejemplos, la estructura de sintaxis puede ser una estructura de sintaxis de predicción de MB o una estructura de sintaxis de predicción de subMB y el elemento de sintaxis puede ser mb_part_vsp_flag o sub_mb_vsp_flag.

[0278] Además, el decodificador de vídeo 30 puede determinar si la unidad de vídeo actual se predice a partir de una imagen VSP (254). Cuando la unidad de vídeo actual no se prediga a partir de una imagen VSP ("NO" de 254), el decodificador de vídeo 30 puede decodificar, a partir del flujo de bits, la información de movimiento para la unidad de vídeo actual (256). Además, el decodificador de vídeo 30 puede usar la información de movimiento para la unidad de vídeo actual para reconstruir bloques de muestra de la unidad de vídeo actual (258). Cuando la unidad de vídeo actual se prediga a partir de la imagen VSP ("SÍ" de 256), el decodificador de vídeo 30 puede usar la imagen VSP para reconstruir los bloques de muestra de la unidad de vídeo actual (260).

[0279] La FIG. 5A es un diagrama de flujo que ilustra otro funcionamiento 300 de ejemplo del codificador de vídeo 20, de acuerdo con las técnicas de esta divulgación. Como se ilustra en el ejemplo de la FIG. 5A, el codificador de vídeo 20 puede sintetizar una imagen VSP basándose al menos en parte en un mapa de profundidad y en uno o más componentes de vista de textura (302). Además, el codificador de vídeo 20 puede generar un elemento de sintaxis que indique un modo de VSP de la unidad de vídeo actual (304). El modo de VSP puede indicar si la unidad de vídeo actual se predice a partir de una imagen VSP. Además, el codificador de vídeo 20 puede determinar si la unidad de vídeo actual se predice a partir de la imagen VSP (306).

[0280] Cuando la unidad de vídeo actual no se prediga a partir de la imagen VSP o cuando la unidad de vídeo actual no se prediga a partir de la imagen VSP ("NO" de 306), el codificador de vídeo 20 puede seleccionar, basándose al menos en parte en si una unidad de vídeo vecina se predice a partir de la imagen VSP, un contexto de codificación (308). Un bloque de muestra de la unidad de vídeo vecina está junto a un bloque de muestra de la unidad de vídeo actual. Además, el codificador de vídeo 20 puede usar el contexto de codificación seleccionado para codificar por entropía al menos alguna información de movimiento de la unidad de vídeo actual (310).

[0281] Además, independientemente de si la unidad de vídeo actual se predice a partir de la imagen VSP, el codificador de vídeo 20 puede determinar, basándose al menos en parte en el modo de VSP de la unidad de vídeo actual, un valor de resistencia límite (312). Además, el codificador de vídeo 20 puede realizar, basándose al menos en parte en el valor de resistencia límite, una operación de desbloqueo en un bloque de muestra de la unidad de vídeo actual (314). El codificador de vídeo 20 puede almacenar el bloque de muestra de la unidad de vídeo actual en la memoria intermedia de imágenes decodificadas 116 (316).

[0282] La FIG. 5B es un diagrama de flujo que ilustra otra operación 350 de ejemplo del decodificador de vídeo 30, de acuerdo con las técnicas de esta divulgación. Como se ilustra en el ejemplo de la FIG. 5B, el decodificador de vídeo 30 puede sintetizar una imagen VSP basándose al menos en parte en un mapa de profundidad y uno o más componentes de vista de textura (352). Además, el decodificador de vídeo 30 puede decodificar, a partir de un flujo de bits, un elemento de sintaxis que indique un modo de VSP de una unidad de vídeo actual (354). El modo de VSP puede indicar si la unidad de vídeo actual se predice a partir de una imagen VSP. Además, el decodificador de vídeo 30 puede determinar, basándose al menos en parte en el elemento de sintaxis, si la unidad de vídeo actual se predice a partir de la imagen VSP (356).

[0283] Cuando la unidad de vídeo actual no se prediga a partir de la imagen VSP ("NO" de 356), el decodificador de vídeo 30 puede seleccionar, basándose al menos en parte en si una unidad de vídeo vecina se predice a partir de la imagen VSP, un contexto de codificación (358) Un bloque de muestra de la unidad de vídeo vecina está junto a un bloque de muestra de la unidad de vídeo actual. Además, el decodificador de vídeo 30 puede usar el contexto de codificación seleccionado para decodificar por entropía al menos alguna información de movimiento de la unidad de vídeo actual (360).

[0284] Cuando la unidad de vídeo actual no se prediga a partir de la imagen VSP o cuando la unidad de vídeo actual se prediga a partir de la imagen VSP ("SÍ" de 356), el decodificador de vídeo 30 puede determinar, basándose al menos en parte en el modo de VSP de la unidad de vídeo actual, un valor de resistencia límite (362). Además, el decodificador de vídeo 30 puede realizar, basándose al menos en parte en el valor de resistencia límite, una operación de desbloqueo en un bloque de muestra de la unidad de vídeo actual (364). El decodificador de vídeo 30 puede almacenar el bloque de muestra de la unidad de vídeo actual en la memoria intermedia de imagen decodificada 162 (366).

[0285] La FIG. 6A es un diagrama de flujo que ilustra otro funcionamiento 400 de ejemplo del codificador de vídeo 20, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 6A, el codificador de vídeo 20 puede indicar, en el flujo de bits, una estructura de sintaxis de cabecera de fragmento para un fragmento que incluya un MB actual (402). La estructura de sintaxis de cabecera de fragmento puede incluir un elemento de sintaxis (por ejemplo, slice_vsp_flag) que indique si se permite la VSP para el fragmento.

[0286] Además, el codificador de vídeo 20 puede determinar si se permite la VSP para el fragmento y si el MB actual se codifica usando el modo de omisión (404). Cuando la VSP se permita para el fragmento y el MB actual se codifique usando el modo de omisión ("SÍ" de 404), el codificador de vídeo 20 puede incluir, en una estructura de sintaxis de datos de fragmento para el fragmento, un elemento de sintaxis (por ejemplo, skip_from_vsp_flag) que indique si el MB actual se predice a partir de la imagen VSP o si el MB actual se predice a partir de otra imagen de referencia (406). De lo contrario, cuando el VSP no se permita para el fragmento o el MB actual no se codifique usando el modo de omisión ("NO" de 404), el codificador de vídeo 20 puede omitir el elemento de sintaxis (por ejemplo, skip_from_vsp_flag) de la estructura de sintaxis de fragmento de datos de fragmento (408). En cualquier caso, el codificador de vídeo 20 puede señalar, en el flujo de bits, la estructura de sintaxis de cabecera de fragmento y la estructura de sintaxis de datos de fragmento (410). De esta forma, el codificador de vídeo 20 puede señalar, en el flujo de bits, un elemento de sintaxis de estructura de sintaxis de datos de fragmento para un fragmento que incluya el MB, en el que la estructura de sintaxis de datos de fragmento incluya un segundo elemento de sintaxis cuando el MB se codifique usando el modo de omisión, indicando el segundo elemento de sintaxis si todos los MB se predicen a partir de la imagen VSP o si el MB se predice a partir de otra imagen de referencia.

[0287] La FIG. 6B es un diagrama de flujo que ilustra otra operación 450 de ejemplo del decodificador de vídeo 30, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 6B, el decodificador de vídeo 30 puede decodificar, a partir del flujo de bits, una estructura de sintaxis de cabecera de fragmento para un fragmento que incluya un MB actual (452). La estructura de sintaxis de cabecera de fragmento puede incluir un elemento de sintaxis (por ejemplo, slice_vsp_flag) que indique si la VSP se permite para el fragmento.

[0288] Además, el decodificador de vídeo 30 puede determinar si la VSP se permite para el fragmento y si el MB actual se codifica usando el modo de omisión (454). Cuando la VSP se permita para el fragmento y el MB actual se codifique usando el modo de omisión ("SÍ" de 454), el decodificador de vídeo 30 puede determinar que una estructura de sintaxis de fragmento de datos para el fragmento incluye un elemento de sintaxis (por ejemplo, skip_from_vsp_flag) que indica si el MB actual se predice a partir de la imagen VSP o si el MB actual se predice a partir de otra imagen de referencia (456). De lo contrario, cuando la VSP no se permita para el fragmento o el MB actual no se codifique usando el modo de omisión ("NO" de 454), el decodificador 30 puede determinar que el elemento de sintaxis (por ejemplo, skip_from_vsp_flag) se omite de la estructura de sintaxis de datos de fragmento (458) En cualquier caso, el decodificador de vídeo 30 puede reconstruir bloques de muestra del MB actual (460). De esta manera, el decodificador de vídeo 30 puede decodificar, del flujo de bits, un elemento de sintaxis de estructura de sintaxis de datos de fragmento para un fragmento que incluya el MB, en el que la estructura de sintaxis de datos de fragmento incluya un segundo elemento de sintaxis cuando el MB se codifique usando el modo de omisión, indicando el segundo elemento de sintaxis si todos los MB se predicen a partir de la imagen VSP o si el MB se predice a partir de otra imagen de referencia.

[0289] La FIG. 7A es un diagrama de flujo que ilustra otra operación 500 de ejemplo del codificador de vídeo 20, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 7A, el codificador de vídeo 20 puede determinar si un MB vecino superior y un MB vecino izquierdo están disponibles para su uso en la codificación de un MB actual, si el MB vecino anterior y el MB vecino izquierdo se predicen a partir de la imagen VSP y si existe la imagen VSP (502)

[0290] Cuando el MB vecino anterior y el MB vecino izquierdo están disponibles para su uso en la codificación del MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y existe la imagen VSP ("SÍ" de 502), el codificador de vídeo 20 puede incluir, en una estructura de sintaxis de datos de fragmento, un elemento de sintaxis que indique que el MB actual se predice a partir de la imagen VSP y que no se señala ningún residuo para el MB actual (504). De lo contrario ("NO" de 502), el codificador de vídeo 20 puede omitirse del elemento de sintaxis de la estructura de sintaxis de datos de fragmento (506). En cualquier caso, el codificador de vídeo 20 puede indicar, en el flujo de bits, la estructura de sintaxis de datos de fragmento (508). De esta forma, el codificador de vídeo 20 puede señalar, en el flujo de bits, una estructura de sintaxis de datos de fragmento, incluyendo la estructura de sintaxis de datos de fragmento un segundo elemento de sintaxis cuando un MB vecino superior y un MB vecino izquierdo estén disponibles para codificar la unidad de vídeo actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y existe la imagen VSP, y el segundo elemento de sintaxis indica que el MB se predice a partir de la imagen VSP y que no se señala ningún residuo para el MB.

[0291] La FIG. 7B es un diagrama de flujo que ilustra otra operación 550 de ejemplo del decodificador de vídeo 30, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 7B, el decodificador de vídeo 30 puede determinar si un MB vecino superior y un MB vecino izquierdo están disponibles para su uso en la codificación de un MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y existe la imagen VSP (552).

[0292] Cuando el MB vecino anterior y el MB vecino izquierdo están disponibles para su uso en la codificación del MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y existe la imagen VSP ("SÍ" de 552), el decodificador de vídeo 30 puede determinar que una estructura de sintaxis de datos de fragmento incluye un elemento de sintaxis que indica que el MB actual se predice a partir de la imagen VSP y que no se señala ningún residuo para el MB actual (554). De lo contrario ("NO" de 552), el decodificador de vídeo 30 puede determinar que el elemento de sintaxis se omite de la estructura de sintaxis de datos de fragmento (556). En cualquier caso, el decodificador de vídeo 30 puede reconstruir bloques de muestra del MB actual (558). De esta forma, el codificador de vídeo 30 puede decodificar, a partir del flujo de bits, una estructura de sintaxis de datos de fragmento, incluyendo la estructura de sintaxis de datos de fragmento un elemento de sintaxis cuando un MB vecino superior y un MB vecino izquierdo estén disponibles para su uso en la codificación de la unidad de vídeo actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y existe la imagen VSP, indicando el elemento de sintaxis que el MB se predice a partir de la imagen VSP y que no se señala ningún residuo para el MB.

[0293] La FIG. 8A es un diagrama de flujo que ilustra otra operación 600 de ejemplo del codificador de vídeo 20, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 8A, el codificador de vídeo 20 puede determinar si un MB vecino superior y un MB vecino izquierdo están disponibles para su uso en la codificación de un MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y el MB actual no se predice de la imagen VSP (602). Cuando el MB vecino superior y el MB vecino izquierdo están disponibles para su uso en la codificación del MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y el MB actual no se predice a partir de la imagen VSP ("SÍ "de 602), el codificador de vídeo 20 puede incluir, en una estructura de sintaxis de capa de MB para el MB actual, un elemento de sintaxis que indique un tipo del MB actual (604). De lo contrario ("NO" de 602), el codificador de vídeo 20 puede omitirse del elemento de sintaxis de la estructura de sintaxis de capa de MB (606).

[0294] Además, el codificador de vídeo 20 puede determinar si el MB actual se predice a partir de la imagen VSP (608). Cuando el MB actual no se prediga a partir de la imagen VSP ("NO" de 608), el codificador de vídeo 20 puede incluir, en la estructura de sintaxis de capa de MB, una estructura de sintaxis de predicción de subMB o una estructura de sintaxis de predicción de MB (610). De lo contrario ("SÍ" de 608), el codificador de vídeo 20 puede omitir cualquier estructura de sintaxis de predicción de subMB y cualquier estructura de sintaxis de predicción de MB de la estructura de sintaxis de capa de MB (612). En cualquier caso, el codificador de vídeo 20 puede indicar, en el flujo de bits, la estructura de sintaxis de capa de MB (614).

[0295] De esta forma, el codificador de vídeo 20 puede señalar, en el flujo de bits, una estructura de sintaxis de capa de MB para el MB. La estructura de sintaxis de capa de MB puede incluir un segundo elemento de sintaxis cuando un MB vecino superior y un MB vecino izquierdo estén disponibles para su uso en la codificación de la unidad de vídeo actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y el MB no se predice a partir de la imagen VSP, indicando el segundo elemento de sintaxis un tipo de MB. La estructura de sintaxis de capa de MB puede incluir una estructura de sintaxis de predicción de subMB o una estructura de sintaxis de predicción de MB cuando el MB no se prediga a partir de la imagen VSP.

[0296] La FIG. 8B es un diagrama de flujo que ilustra otra operación 650 de ejemplo del decodificador de vídeo 30, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 8B, el decodificador de vídeo 30 puede determinar si un MB vecino superior y un MB vecino izquierdo están disponibles para su uso en la codificación de un MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y el MB actual no se predice a partir de la imagen VSP (652). Cuando el MB vecino superior y el MB vecino izquierdo están disponibles para su uso en la codificación del MB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y el MB actual no se predice a partir de la imagen VSP ("SÍ "de 652), el decodificador de vídeo 30 puede determinar que una estructura de sintaxis de capa de MB para el MB actual incluye un elemento de sintaxis que indica un tipo del MB actual (654). De lo contrario ("NO" de 652), el decodificador de vídeo 30 puede determinar que el elemento de sintaxis se omite de la estructura de sintaxis de capa de MB (656).

[0297] Además, el decodificador de vídeo 30 puede determinar si el MB actual se predice a partir de la imagen VSP (658). Cuando el MB actual no se predice a partir de la imagen VSP ("NO" de 658), el decodificador de vídeo 30 puede determinar que la estructura de sintaxis de capa de MB incluye una estructura de sintaxis de predicción de subMB o una estructura de sintaxis de predicción de MB (660). De lo contrario ("SÍ" de 658), el decodificador de vídeo 30 puede determinar que cualquier estructura de sintaxis de predicción de subMB y cualquier estructura de sintaxis de predicción de MB se omiten de la estructura de sintaxis de capa de MB (662). En cualquier caso, el decodificador de vídeo 30 puede reconstruir bloques de muestra del MB (664).

[0298] De esta manera, el decodificador de vídeo 30 puede decodificar, a partir del flujo de bits, una estructura de sintaxis de capa de MB para el MB, incluyendo la estructura de sintaxis de capa de MB un segundo elemento de sintaxis cuando un MB vecino superior y un MB vecino izquierdo estén disponibles para su uso al codificar la unidad de vídeo actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y el MB no se predice a partir de la imagen VSP, indicando el segundo elemento de sintaxis un tipo del MB, e incluyendo la estructura de sintaxis de capa de MB una estructura de sintaxis de predicción de subMB o una estructura de sintaxis de predicción de MB cuando el MB no se prediga a partir de la imagen VSP.

[0299] La FIG. 9A es un diagrama de flujo que ilustra otra operación 700 de ejemplo del codificador de vídeo 20, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 9A, el codificador de vídeo 20 puede determinar, si un MB vecino superior y un MB vecino izquierdo están disponibles para su uso en la codificación de una partición actual de subMB, si el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP y si la partición de subMB actual no se predice a partir de la imagen VSP (702).

[0300] Cuando el MB vecino superior y el MB vecino izquierdo están disponibles para su uso en la codificación de la partición de subMB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y la partición de subMB actual no se predice a partir de la imagen VSP ("SÍ" de 702), el codificador de vídeo 20 puede incluir, en una estructura de sintaxis de predicción de subMB para la partición actual de subMB, un elemento de sintaxis que indique un tipo de partición de subMB (704). De lo contrario ("NO" de 702), el codificador de vídeo 20 puede omitir el elemento de sintaxis de la estructura de sintaxis de predicción de subMB (706). En cualquier caso, el codificador de vídeo 20 puede indicar, en el flujo de bits, la estructura de sintaxis de predicción de subMB (708). De esta forma, el codificador de vídeo 20 puede señalar, en el flujo de bits, una estructura de sintaxis de predicción de subMB para la partición de subMB, incluyendo la estructura de sintaxis de predicción de subMB un segundo elemento de sintaxis cuando un MB vecino superior y un MB vecino izquierdo estén disponibles para su uso en la codificación de la unidad de vídeo actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y la partición de subMB no se predice a partir de la imagen VSP.

[0301] La FIG. 9B es un diagrama de flujo que ilustra otra operación 750 de ejemplo del decodificador de vídeo 30, de acuerdo con las técnicas de esta divulgación. En el ejemplo de la FIG. 9B, el decodificador de vídeo 30 puede determinar si un MB vecino superior y un MB vecino izquierdo están disponibles para su uso en la codificación de una partición de subMB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y una partición de subMB actual no se predice a partir de la imagen VSP (752).

[0302] Cuando el MB vecino superior y el MB vecino izquierdo están disponibles para su uso en la codificación de la partición de subMB actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y la partición de subMB actual no se predice a partir de la imagen VSP ("SÍ" de 752), el decodificador de vídeo 30 puede determinar que una estructura de sintaxis de predicción de subMB para la partición de subMB actual incluye un elemento de sintaxis que indica un tipo de la partición de subMB (754) De lo contrario ("NO" de 752), el decodificador de vídeo 30 puede determinar que el elemento de sintaxis se omite de la estructura de sintaxis de predicción de subMB (756). En cualquier caso, el decodificador de vídeo 30 puede reconstruir bloques de muestra de la partición de subMB (758). De esta manera, el decodificador de vídeo 30 puede decodificar, desde el flujo de bits, una estructura de sintaxis de predicción de subMB para la partición de subMB, incluyendo la estructura de sintaxis de predicción de subMB un segundo elemento de sintaxis cuando un MB vecino superior y un MB vecino izquierdo estén disponibles para su uso en la codificación de la unidad de vídeo actual, el MB vecino superior y el MB vecino izquierdo se predicen a partir de la imagen VSP, y la partición de subMB no se predice a partir de la imagen VSP, el segundo elemento de sintaxis indica un tipo de partición de subMB.

[0303] La FIG. 10 es un diagrama conceptual que ilustra una orden de decodificación de 3DV de ejemplo. La orden de decodificación de MVC puede ser una orden de flujo de bits. En el ejemplo de la FIG. 10, cada cuadrado corresponde a un componente de vista. Las columnas de cuadrados corresponden a las unidades de acceso. Cada unidad de acceso puede definirse para contener las imágenes codificadas de todas las visualizaciones de una instancia de tiempo. Las filas de cuadrados corresponden a las vistas. En el ejemplo de la FIG. 10, las unidades de acceso están etiquetadas como T0 ... T7 y las vistas están etiquetadas como S0 ... S7. Debido a que cada componente de vista de una unidad de acceso se decodifica antes de cualquier componente de visualización de la siguiente unidad de acceso, el orden de decodificación de la FIG. 10 puede denominarse codificación por primera vez. Como se muestra en el ejemplo de la FIG. 10, la orden de decodificación de las unidades de acceso puede no ser idéntica a la orden de salida o visualización de las vistas.

[0304] La FIG. 11 es un diagrama conceptual que ilustra un ejemplo de estructura de predicción temporal e inter-vistas. En el ejemplo de la FIG. 11, cada cuadrado corresponde a un componente de vista. Los cuadrados etiquetados como "I" son componentes de vista sometidos a intrapredicción. Los cuadrados etiquetados como "P" son componentes de vista sometidos a interpredicción unidireccionalmente. Los cuadrados etiquetados como "B" y "b" son componentes de vista sometidos a interpredicción bidireccionalmente. Los cuadrados etiquetados como "b" pueden usar cuadrados etiquetados como "B" como imágenes de referencia. Una flecha que apunta desde un primer cuadrado a un segundo cuadrado indica que el primer cuadrado está disponible en la interpredicción como imagen de referencia para el segundo cuadrado. Como se indica por las flechas verticales en la FIG. 11, los componentes de vista en diferentes

vistas de la misma unidad de acceso pueden estar disponibles como imágenes de referencia. El uso de un componente de vista de una unidad de acceso como imagen de referencia para otro componente de vista de la misma unidad de acceso puede denominarse predicción inter-vistas.

[0305] En uno o más ejemplos, las funciones descritas pueden implementarse en hardware, software, firmware o en cualquier combinación de los mismos. Si se implementan en software, las funciones pueden almacenarse en o transmitirse a través de, como una o más instrucciones o código, un medio legible por ordenador o ejecutarse mediante una unidad de procesamiento basada en hardware. Los medios legibles por ordenador pueden incluir medios de almacenamiento legibles por ordenador, que correspondan a un medio tangible tal como unos medios de almacenamiento de datos o unos medios de comunicación que incluyan cualquier medio que facilite la transferencia de un programa informático desde un lugar a otro, por ejemplo, de acuerdo con un protocolo de comunicación. De esta manera, los medios legibles por ordenador pueden corresponder en general a (1) unos medios de almacenamiento tangibles legibles por ordenador que sean no transitorios, o (2) un medio de comunicación tal como una señal o una onda portadora. Los medios de almacenamiento de datos pueden ser cualquier medio disponible al que se pueda acceder desde uno o más ordenadores o uno o más procesadores para recuperar instrucciones, código y/o estructuras de datos para la implementación de las técnicas descritas en esta divulgación. Un producto de programa informático puede incluir un medio legible por ordenador.

[0306] A modo de ejemplo, y no de limitación, dichos medios de almacenamiento legibles por ordenador pueden comprender RAM, ROM, EEPROM, CD-ROM u otro almacenamiento de disco óptico, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnético, memoria flash o cualquier otro medio que pueda usarse para almacenar un código de programa deseado en forma de instrucciones o estructuras de datos y al que pueda accederse mediante un ordenador. Además, cualquier conexión recibe debidamente la denominación de medio legible por ordenador. Por ejemplo, si las instrucciones se transmiten desde un sitio web, un servidor u otro origen remoto usando un cable coaxial, un cable de fibra óptica, un par trenzado, una línea de abonado digital (DSL) o tecnologías inalámbricas tales como infrarrojos, radio y microondas, entonces el cable coaxial, el cable de fibra óptica, el par trenzado, la DSL o las tecnologías inalámbricas tales como infrarrojos, radio y microondas se incluyen en la definición de medio. Sin embargo, debería entenderse que los medios de almacenamiento legibles por ordenador y los medios de almacenamiento de datos no incluyen conexiones, ondas portadoras, señales u otros medios transitorios, sino que, en cambio, se orientan a medios de almacenamiento tangibles no transitorios. El término disco, como se usa en el presente documento, incluye un disco compacto (CD), un disco láser, un disco óptico, un disco versátil digital (DVD), un disco flexible y un disco Blu-ray, donde algunos discos habitualmente emiten datos magnéticamente, mientras que otros discos emiten datos ópticamente con láseres. Las combinaciones de lo anterior también deben incluirse dentro del alcance de los medios legibles por ordenador.

[0307] Las instrucciones pueden ejecutarse por uno o más procesadores, tales como uno o más procesadores de señales digitales (DSP), microprocesadores de uso general, circuitos integrados específicos de la aplicación (ASIC), matrices lógicas programables por campo (FPGA) u otros circuitos lógicos equivalentes, integrados o discretos. En consecuencia, el término «procesador», como se usa en el presente documento, puede referirse a cualquiera de las estructuras anteriores o a cualquier otra estructura adecuada para la implementación de las técnicas descritas en el presente documento. Además, en algunos aspectos, la funcionalidad descrita en el presente documento puede proporcionarse dentro de módulos de hardware y/o software dedicados configurados para la codificación y la decodificación, o incorporados en un códec combinado. Asimismo, las técnicas podrían implementarse por completo en uno o más circuitos o elementos lógicos.

[0308] Las técnicas de esta divulgación se pueden implementar en una amplia variedad de dispositivos o aparatos, que incluya un teléfono inalámbrico, un circuito integrado (CI) o un conjunto de CI (por ejemplo, un conjunto de chips). Diversos componentes, módulos o unidades se describen en esta divulgación para enfatizar aspectos funcionales de dispositivos configurados para realizar las técnicas divulgadas, pero no requieren necesariamente su realización mediante diferentes unidades de hardware. En cambio, como se ha descrito anteriormente, diversas unidades pueden combinarse en una unidad de hardware de códec o proporcionarse por medio de un grupo de unidades de hardware interoperativas, incluyendo uno o más procesadores como se describió anteriormente, conjuntamente con software y/o firmware adecuados.

[0309] Se han descrito diversos ejemplos. Estos y otros ejemplos están dentro del alcance de las siguientes reivindicaciones.

REIVINDICACIONES

1. Un procedimiento de decodificación de datos de vídeo en 3D codificados predictivamente, el procedimiento que comprende:

determinar si una unidad de vídeo actual se codifica usando un primer modo, en el que la unidad de vídeo actual es un macrobloque, MB, o una partición de MB de un componente de vista de textura actual de una vista actual de una unidad de acceso actual;

cuando la unidad de vídeo actual se codifica usando el primer modo:

determinar una imagen de vista de referencia para la predicción de síntesis de visión,
en el que la imagen de vista de referencia es la imagen en la unidad de acceso actual en una vista indicada en una cabecera de fragmento como una vista para la síntesis de visión;

establecer un índice de referencia para la unidad de vídeo actual de tal manera que el índice de referencia indique la imagen de vista de referencia; y

derivar un vector de movimiento para la unidad de vídeo actual, en el que el vector de movimiento para la unidad de vídeo actual se refiere a la imagen de vista de referencia y en el que derivar el vector de movimiento comprende:

determinar un valor de profundidad representativo de un bloque de una imagen de profundidad, correspondiendo el bloque de la imagen de profundidad a una unidad de vídeo actual de la imagen de textura correspondiente, en el que el valor de profundidad representativo se determina basándose en un valor máximo, promedio o mediano de píxeles de profundidad predeterminados en el bloque de la imagen de profundidad, en el que los píxeles de profundidad predeterminados son los píxeles de profundidad de esquina en el bloque de la imagen de profundidad;

usar el valor de profundidad representativo para determinar un vector de disparidad; y

establecer el vector de movimiento para la unidad de vídeo actual igual al vector de disparidad; o

cuando la unidad de vídeo actual se codifique usando un segundo modo, decodificar, desde el flujo de bits, información de movimiento para la unidad de vídeo actual;

generar un bloque predictivo para la unidad de vídeo actual basándose en un bloque de referencia indicado por la información de movimiento de la unidad de vídeo actual, incluyendo la información de movimiento de la unidad de vídeo actual el vector de movimiento para la unidad de vídeo actual y el índice de referencia para la unidad de vídeo actual; y

añadir el bloque predictivo a un bloque residual para construir un bloque de muestra de la unidad de vídeo actual.

2. El procedimiento según la reivindicación 1, en el que el primer modo se identifica por uno de los siguientes elementos de sintaxis en 3D-AVC o 3D-HEVC: un sub_mb_vsp_flag, un mb_part_vsp_flag y un vsp_mb_flag.

3. El procedimiento según la reivindicación 1, en el que determinar si la unidad de vídeo actual se codifica usando el primer modo comprende determinar que la unidad de vídeo actual se codifica usando el primer modo cuando un índice de referencia señalizado para la unidad de vídeo actual tenga un valor específico.

4. El procedimiento según la reivindicación 1, en el que establecer el índice de referencia para la unidad de vídeo actual comprende: para cada i de 0 a un número de imágenes de referencia activas en una lista de imágenes de referencia RefPicListO, inclusive, si RefPicList0 [i] es igual a la vista de la imagen de referencia para la predicción de síntesis de visión, estableciendo el índice de referencia en i.

5. Un dispositivo de decodificación de vídeo en 3D (30) para decodificar datos de vídeo en 3D codificados de forma predictiva, el dispositivo que comprende:

medios para determinar si una unidad de vídeo actual se codifica usando un primer modo, en el que la unidad de vídeo actual es un macrobloque, MB, o una partición de MB de un componente de vista de textura actual de una vista actual de una unidad de acceso actual;

medios para determinar, cuando la unidad de vídeo actual se codifique usando el primer modo, una imagen de vista de referencia para la predicción de síntesis de visión, en el que la imagen de referencia sea la

imagen en la unidad de acceso actual en una vista indicada en una cabecera de fragmento como una vista para la síntesis de visión;

medios para establecer, cuando la unidad de vídeo actual se codifique usando el primer modo, un índice de referencia para la unidad de vídeo actual de tal manera que el índice de referencia indique la imagen de vista de referencia;

medios para derivar, cuando la unidad de vídeo actual se codifique usando el primer modo, un vector de movimiento para la unidad de vídeo actual, en el que el vector de movimiento para la unidad de vídeo actual se refiere a la imagen de vista de referencia y en el que derivar el vector de movimiento comprende:

determinar un valor de profundidad representativo de un bloque de una imagen de profundidad, correspondiendo el bloque de la imagen de profundidad a una unidad de vídeo actual de la imagen de textura correspondiente, en el que el valor de profundidad representativo se determina basándose en un valor máximo, promedio o mediano de píxeles de profundidad predeterminados en el bloque de la imagen de profundidad, en el que los píxeles de profundidad predeterminados son los píxeles de profundidad de esquina en el bloque de la imagen de profundidad;

usar el valor de profundidad representativo para determinar un vector de disparidad; y

establecer el vector de movimiento para la unidad de vídeo actual igual al vector de disparidad;

medios para decodificar, cuando la unidad de vídeo actual se codifique usando un segundo modo, desde el flujo de bits, información de movimiento para la unidad de vídeo actual;

medios para generar un bloque predictivo para la unidad de vídeo actual basándose en un bloque de referencia indicado por la información de movimiento de la unidad de vídeo actual, incluyendo la información de movimiento de la unidad de vídeo actual el vector de movimiento para la unidad de vídeo actual y el índice de referencia para la unidad de vídeo actual; y

medios para añadir el bloque predictivo a un bloque residual para construir un bloque de muestra de la unidad de vídeo actual.

6. Un medio de almacenamiento legible por ordenador que tiene instrucciones almacenadas en el mismo que, cuando se ejecuta por uno o más procesadores de un dispositivo de decodificación de vídeo, configura el dispositivo de decodificación de vídeo para llevar a cabo el procedimiento de cualquiera de las reivindicaciones 1 a 4.

7. Un procedimiento para codificar datos de vídeo en 3D, el procedimiento que comprende:

señalar, en un flujo de bits que incluya una representación codificada de múltiples vistas de textura y múltiples vistas de profundidad, si una unidad de vídeo actual se codifica usando un primer modo, en el que la unidad de vídeo actual es un macrobloque, MB, o una partición de MB de un componente de vista de textura actual de una vista actual de una unidad de acceso actual;

cuando la unidad de vídeo actual se codifique usando un segundo modo y no el primer modo, señalar, en el flujo de bits, un índice de referencia para la unidad de vídeo actual y la información de movimiento para la unidad de vídeo actual;

cuando la unidad de vídeo actual se codifique usando el primer modo:

determinar una imagen de vista de referencia para la predicción de síntesis de visión, en el que la imagen de vista de referencia sea la imagen en la unidad de acceso actual en una vista indicada en una cabecera de fragmento como una vista para la síntesis de visión;

establecer el índice de referencia para la unidad de vídeo actual de tal manera que el índice de referencia indique la imagen de vista de referencia;

derivar el vector de movimiento para la unidad de vídeo actual, en el que el vector de movimiento para la unidad de vídeo actual se refiere a la imagen de vista de referencia y en el que derivar el vector de movimiento comprende:

determinar un valor de profundidad representativo de un bloque de una imagen de profundidad, correspondiendo el bloque de la imagen de profundidad a una unidad de vídeo actual de la imagen de textura correspondiente, en el que el valor de profundidad representativo se determina basándose en un valor máximo, promedio o mediano de los píxeles de profundidad predeterminados en el

bloque de la imagen de profundidad, en el que los píxeles de profundidad predeterminados son los píxeles de profundidad de esquina en el bloque de la imagen de profundidad;

usar el valor de profundidad representativo para determinar un vector de disparidad; y

establecer el vector de movimiento para la unidad de vídeo actual igual al vector de disparidad; y

omitir (208), del flujo de bits, el vector de movimiento para la unidad de vídeo actual; y

emitir el flujo de bits.

8. El procedimiento según la reivindicación 7, en el que el primer modo de VSP se identifica por uno de los siguientes elementos de sintaxis en 3D-AVC o 3D-HEVC: un sub_mb_vsp_flag, un mb_part_vsp_flag y un vsp_mb_flag.

9. El procedimiento según la reivindicación 7, en el que señalar si la unidad de vídeo actual se codifica usando el primer modo de VSP comprende señalar, en el flujo de bits, que el índice de referencia para la unidad de vídeo actual tiene un valor específico.

10. El procedimiento según la reivindicación 7, en el que establecer el índice de referencia para la unidad de vídeo actual comprende: para cada i de 0 a un número de imágenes de referencia activas en una lista de imágenes de referencia RefPicListO, inclusive, si RefPicList0 [i] es igual a la imagen de vista de referencia para la predicción de síntesis de visión, estableciendo el índice de referencia en i.

11. Un dispositivo de codificación de vídeo en 3D (20) que comprende:

medios para señalar, en un flujo de bits que incluya una representación codificada de múltiples vistas de textura y múltiples vistas de profundidad, si una unidad de vídeo actual se codifica usando un primer modo, en el que la unidad de vídeo actual es un macrobloque, MB, o una partición de MB de un componente de vista de textura actual de una vista actual de una unidad de acceso actual;

medios para señalar, cuando la unidad de vídeo actual se codifique usando un segundo modo y no el primer modo, en el flujo de bits, información de movimiento para la unidad de vídeo actual;

medios para determinar, cuando la unidad de vídeo actual se codifique usando el primer modo, una imagen de referencia de referencia para la predicción de síntesis de visión, en el que la imagen de vista de referencia sea la imagen en la unidad de acceso actual en una vista indicada en una cabecera de fragmento como una vista para la síntesis de visión;

medios para establecer, cuando la unidad de vídeo actual se codifique usando el primer modo, el índice de referencia para la unidad de vídeo actual de tal manera que el índice de referencia indique la imagen de vista de referencia;

medios para derivar, cuando la unidad de vídeo actual se codifique usando el primer modo, el vector de movimiento para la unidad de vídeo actual, en el que el vector de movimiento para la unidad de vídeo actual se refiere a la imagen de vista de referencia y en el que derivar el vector de movimiento para la unidad de vídeo actual comprende:

determinar un valor de profundidad representativo de un bloque de una imagen de profundidad, correspondiendo el bloque de la imagen de profundidad a una unidad de vídeo actual de la imagen de textura correspondiente, en el que el valor de profundidad representativo se determina basándose en un valor máximo, promedio o mediano de píxeles de profundidad predeterminados en el bloque de la imagen de profundidad, en el que los píxeles de profundidad predeterminados son los píxeles de profundidad de esquina en el bloque de la imagen de profundidad;

usar el valor de profundidad representativo para generar un vector de disparidad; y

establecer el valor del vector de movimiento para la unidad de vídeo actual en el valor del vector de disparidad; y

medios para omitir, cuando la unidad de vídeo actual se codifique usando el modo de VSP unificado, del flujo de bits, la información de movimiento para la unidad de vídeo actual; y

medios para emitir el flujo de bits.

12. Un medio de almacenamiento legible por ordenador que tiene instrucciones almacenadas en el mismo que, cuando se ejecutan por uno o más procesadores de un dispositivo de codificación de vídeo, configura el dispositivo de codificación de vídeo para llevar a cabo el procedimiento de cualquiera de las reivindicaciones 7 a 10.

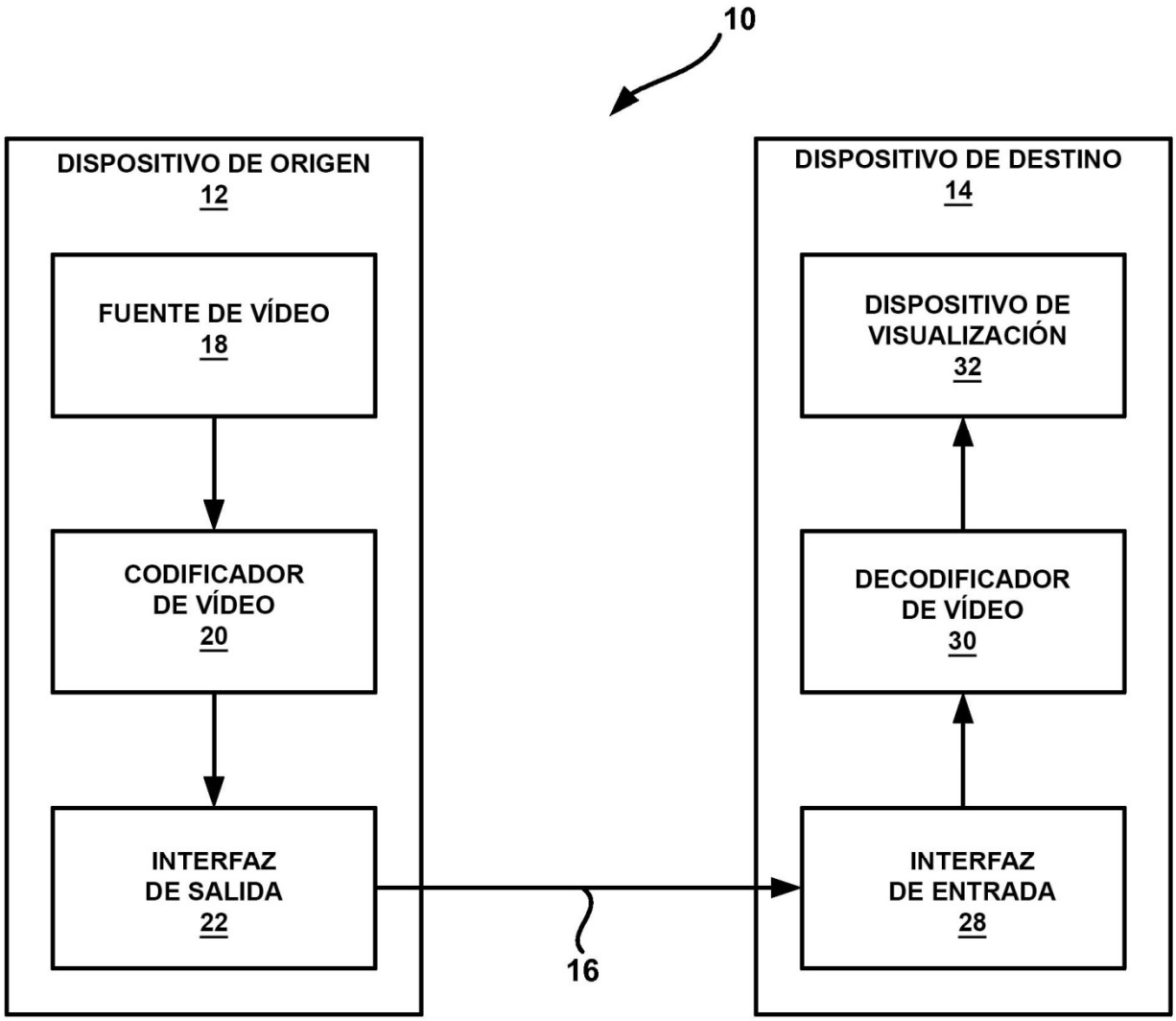
10
DISPOSITIVO DE ORIGEN
12
FUENTE DE VÍDEO
18
CODIFICADOR
DE VÍDEO
20
INTERFAZ
DE SALIDA
22
16
DISPOSITIVO DE DESTINO
14
DISPOSITIVO DE
VISUALIZACIÓN
32
DECODIFICADOR
DE VÍDEO
30
INTERFAZ
DE ENTRADA
28

FIG. 1

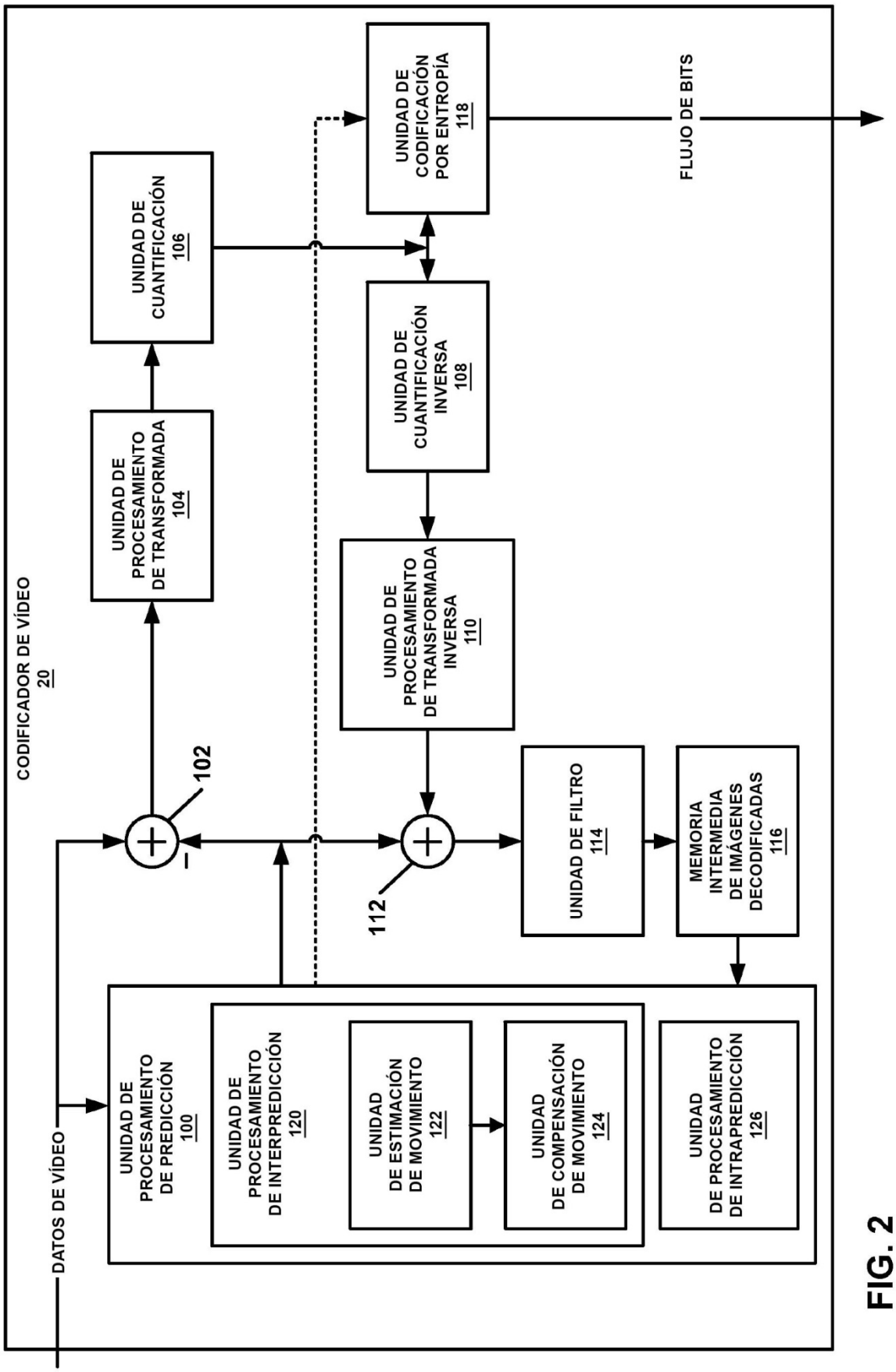
CODIFICADOR DE VÍDEO
20
DATOS DE VÍDEO
UNIDAD DE PROCESAMIENTO DE PREDICCIÓN
100
UNIDAD DE PROCESAMIENTO DE INTERPREDICCIÓN
120
UNIDAD DE ESTIMACIÓN DE MOVIMIENTO
122
UNIDAD DE COMPENSACIÓN DE MOVIMIENTO
124
UNIDAD DE PROCESAMIENTO DE INTRAPREDICCIÓN
126
102
112
UNIDAD DE PROCESAMIENTO DE TRANSFORMADA
104
UNIDAD DE CUANTIFICACIÓN
106
UNIDAD DE CUANTIFICACIÓN INVERSA
108
UNIDAD DE PROCESAMIENTO DE TRANSFORMADA INVERSA
110
UNIDAD DE FILTRO
114
MEMORIA INTERMEDIA DE IMÁGENES DECODIFICADAS
116
UNIDAD DE CODIFICACIÓN POR ENTROPÍA
118
FLUJO DE BITS

FIG. 2

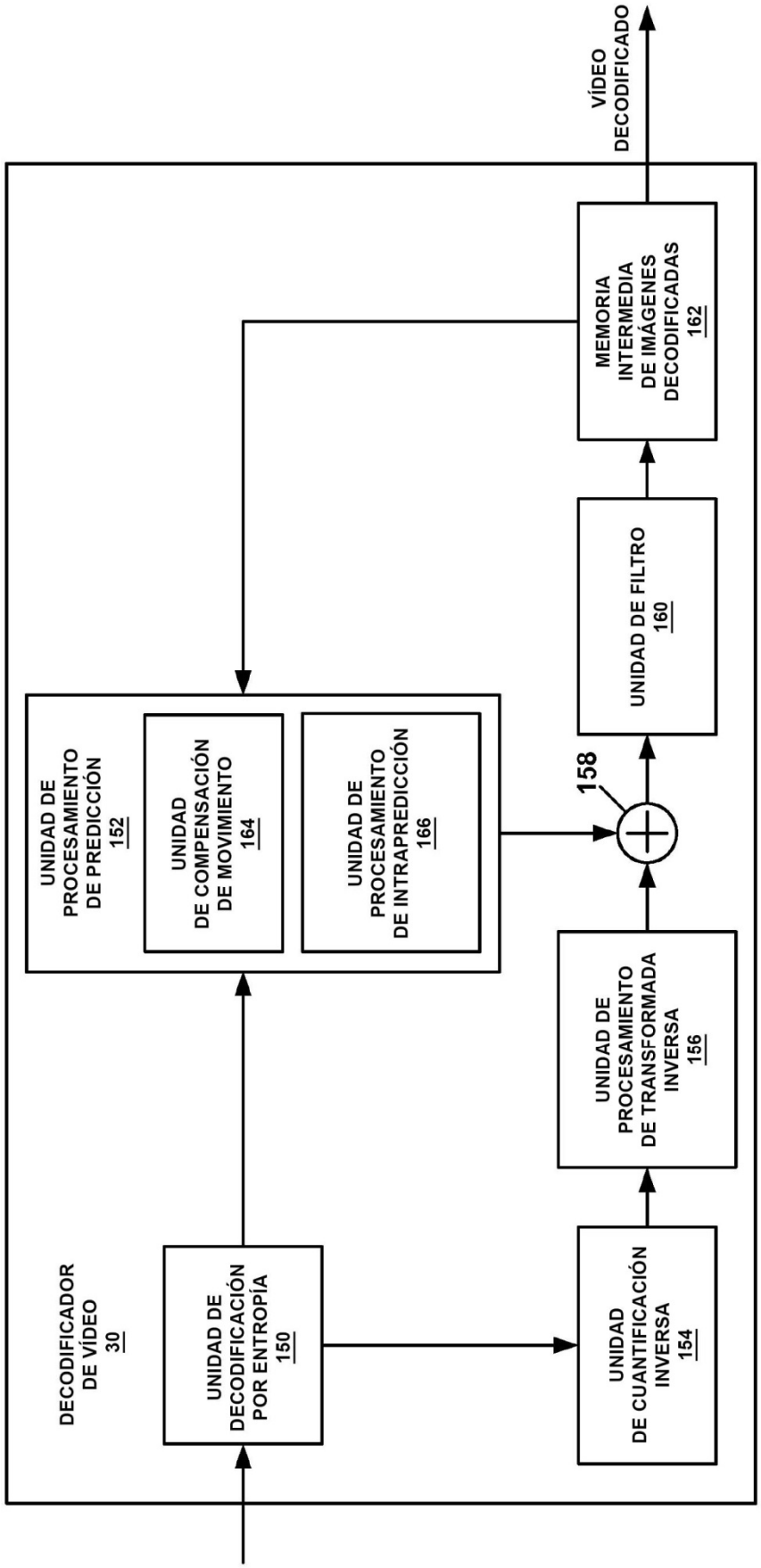
DECODIFICADOR DE VÍDEO 30
UNIDAD DE DECODIFICACIÓN POR ENTROPÍA 150
UNIDAD DE PROCESAMIENTO DE PREDICCIÓN 152
UNIDAD DE COMPENSACIÓN DE MOVIMIENTO 164
UNIDAD DE PROCESAMIENTO DE INTRAPREDICCIÓN 166
UNIDAD DE CUANTIFICACIÓN INVERSA 154
UNIDAD DE PROCESAMIENTO DE TRANSFORMADA INVERSA 156
158
UNIDAD DE FILTRO 160
MEMORIA INTERMEDIA DE IMÁGENES DECODIFICADAS 162
VÍDEO DECODIFICADO

FIG. 3

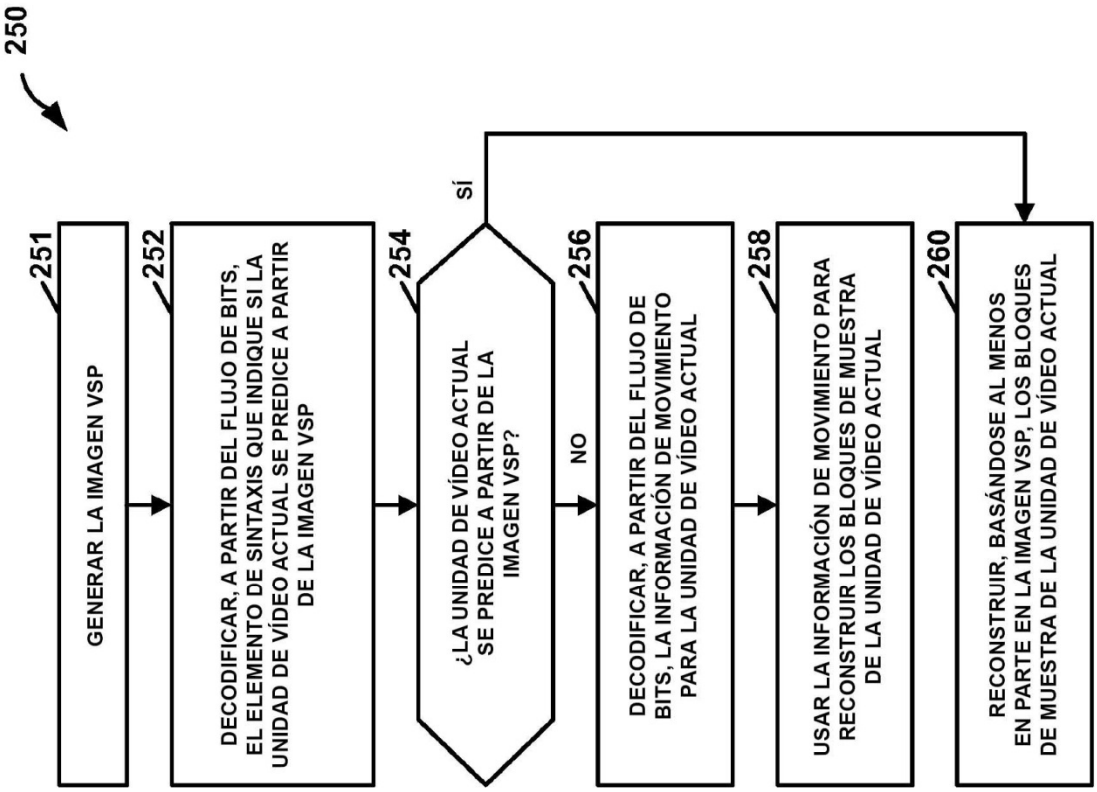
250
251
GENERAR LA IMAGEN VSP
252
DECODIFICAR, A PARTIR DEL FLUJO DE BITS, EL ELEMENTO DE SINTAXIS QUE INDIQUE SI LA UNIDAD DE VÍDEO ACTUAL SE PREDICE A PARTIR DE LA IMAGEN VSP
254
¿LA UNIDAD DE VÍDEO ACTUAL SE PREDICE A PARTIR DE LA IMAGEN VSP?
sí
NO
256
DECODIFICAR, A PARTIR DEL FLUJO DE BITS, LA INFORMACIÓN DE MOVIMIENTO PARA LA UNIDAD DE VÍDEO ACTUAL
258
USAR LA INFORMACIÓN DE MOVIMIENTO PARA RECONSTRUIR LOS BLOQUES DE MUESTRA DE LA UNIDAD DE VÍDEO ACTUAL
260
RECONSTRUIR, BASÁNDOSE AL MENOS EN PARTE EN LA IMAGEN VSP, LOS BLOQUES DE MUESTRA DE LA UNIDAD DE VÍDEO ACTUAL

FIG. 4B

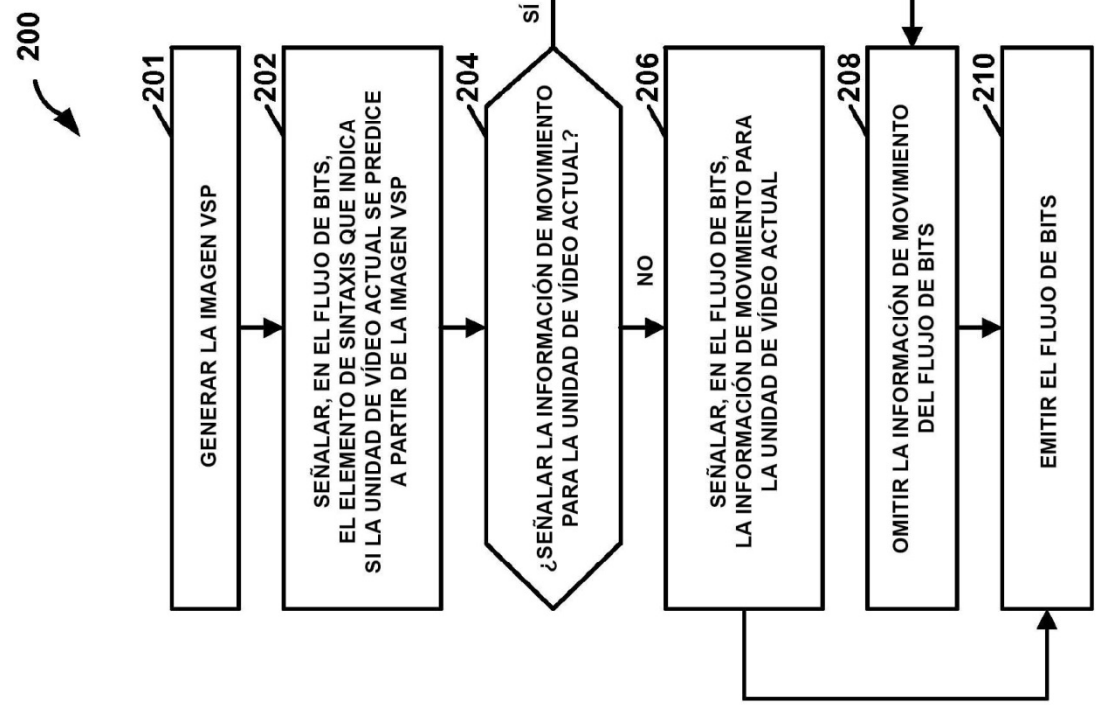
200
201
GENERAR LA IMAGEN VSP
202
SEÑALAR, EN EL FLUJO DE BITS, EL ELEMENTO DE SINTAXIS QUE INDICA SI LA UNIDAD DE VÍDEO ACTUAL SE PREDICE A PARTIR DE LA IMAGEN VSP
204
¿SEÑALAR LA INFORMACIÓN DE MOVIMIENTO PARA LA UNIDAD DE VÍDEO ACTUAL?
sí
NO
206
SEÑALAR, EN EL FLUJO DE BITS, LA INFORMACIÓN DE MOVIMIENTO PARA LA UNIDAD DE VÍDEO ACTUAL
208
OMITIR LA INFORMACIÓN DE MOVIMIENTO DEL FLUJO DE BITS
210
EMITIR EL FLUJO DE BITS

FIG. 4A

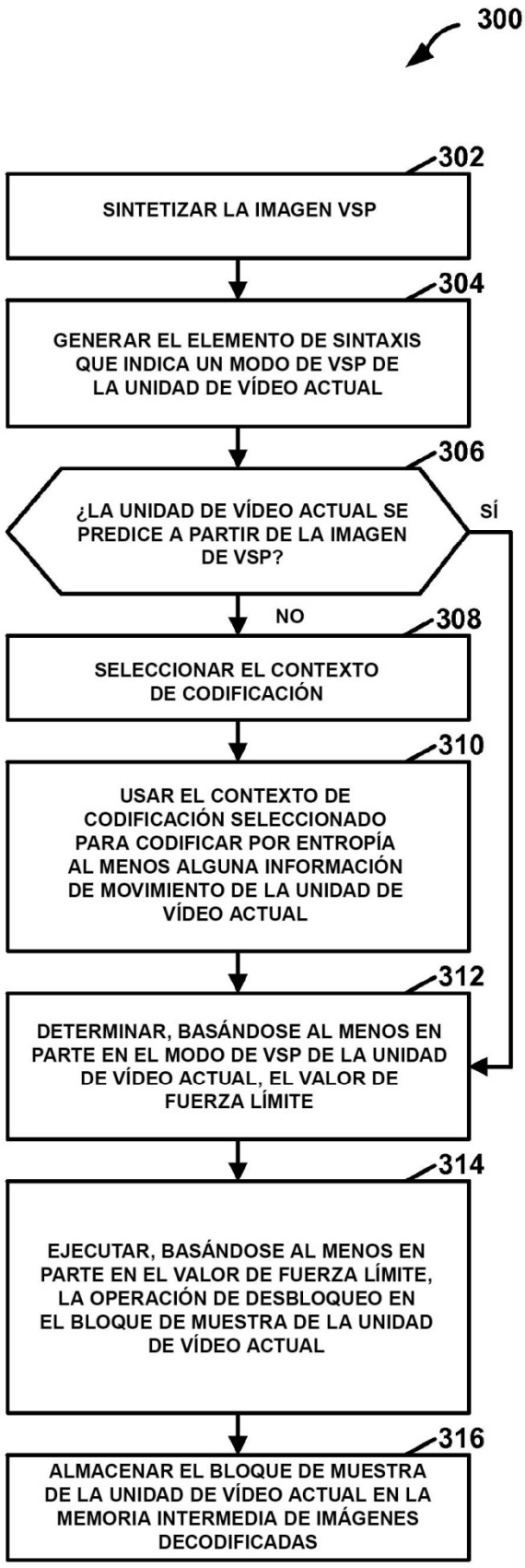
300
302
SINTETIZAR LA IMAGEN VSP
304
GENERAR EL ELEMENTO DE SINTAXIS QUE INDICA UN MODO DE VSP DE LA UNIDAD DE VÍDEO ACTUAL
306
¿LA UNIDAD DE VÍDEO ACTUAL SE PREDICE A PARTIR DE LA IMAGEN DE VSP?
SÍ
NO
308
SELECCIONAR EL CONTEXTO DE CODIFICACIÓN
310
USAR EL CONTEXTO DE CODIFICACIÓN SELECCIONADO PARA CODIFICAR POR ENTROPÍA AL MENOS ALGUNA INFORMACIÓN DE MOVIMIENTO DE LA UNIDAD DE VÍDEO ACTUAL
312
DETERMINAR, BASÁNDOSE AL MENOS EN PARTE EN EL MODO DE VSP DE LA UNIDAD DE VÍDEO ACTUAL, EL VALOR DE FUERZA LÍMITE
314
EJECUTAR, BASÁNDOSE AL MENOS EN PARTE EN EL VALOR DE FUERZA LÍMITE, LA OPERACIÓN DE DESBLOQUEO EN EL BLOQUE DE MUESTRA DE LA UNIDAD DE VÍDEO ACTUAL
316
ALMACENAR EL BLOQUE DE MUESTRA DE LA UNIDAD DE VÍDEO ACTUAL EN LA MEMORIA INTERMEDIA DE IMÁGENES DECODIFICADAS

FIG. 5A

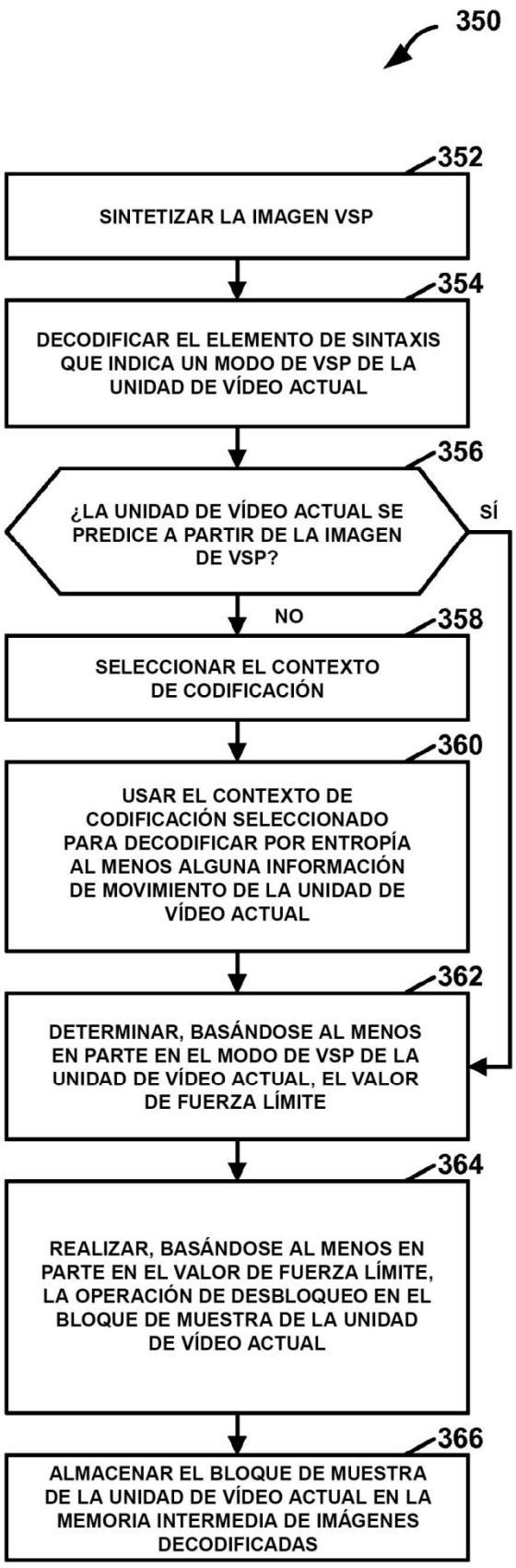
350
352
SINTETIZAR LA IMAGEN VSP
354
DECODIFICAR EL ELEMENTO DE SINTAXIS QUE INDICA UN MODO DE VSP DE LA UNIDAD DE VÍDEO ACTUAL
356
¿LA UNIDAD DE VÍDEO ACTUAL SE PREDICE A PARTIR DE LA IMAGEN DE VSP?
SÍ
NO
358
SELECCIONAR EL CONTEXTO DE CODIFICACIÓN
360
USAR EL CONTEXTO DE CODIFICACIÓN SELECCIONADO PARA DECODIFICAR POR ENTROPÍA AL MENOS ALGUNA INFORMACIÓN DE MOVIMIENTO DE LA UNIDAD DE VÍDEO ACTUAL
362
DETERMINAR, BASÁNDOSE AL MENOS EN PARTE EN EL MODO DE VSP DE LA UNIDAD DE VÍDEO ACTUAL, EL VALOR DE FUERZA LÍMITE
364
REALIZAR, BASÁNDOSE AL MENOS EN PARTE EN EL VALOR DE FUERZA LÍMITE, LA OPERACIÓN DE DESBLOQUEO EN EL BLOQUE DE MUESTRA DE LA UNIDAD DE VÍDEO ACTUAL
366
ALMACENAR EL BLOQUE DE MUESTRA DE LA UNIDAD DE VÍDEO ACTUAL EN LA MEMORIA INTERMEDIA DE IMÁGENES DECODIFICADAS

FIG. 5B

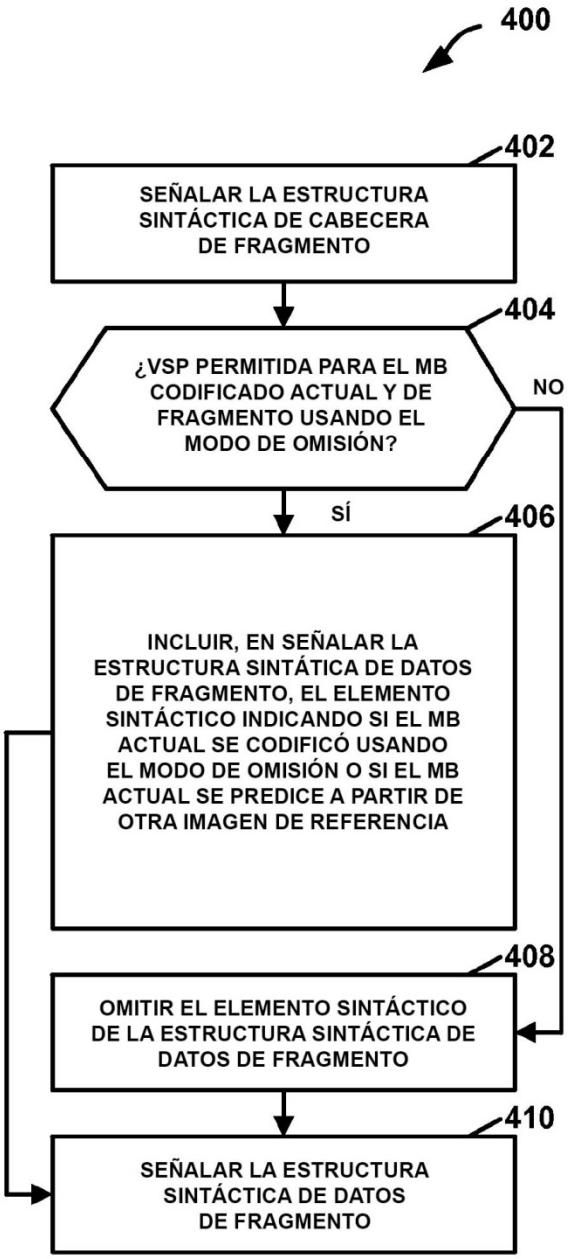
400
402
SEÑALAR LA ESTRUCTURA SINTÁCTICA DE CABECERA DE FRAGMENTO
404
¿VSP PERMITIDA PARA EL MB CODIFICADO ACTUAL Y DE FRAGMENTO USANDO EL MODO DE OMISIÓN?
NO
SÍ
406
INCLUIR, EN SEÑALAR LA ESTRUCTURA SINTÁTICA DE DATOS DE FRAGMENTO, EL ELEMENTO SINTÁCTICO INDICANDO SI EL MB ACTUAL SE CODIFICÓ USANDO EL MODO DE OMISIÓN O SI EL MB ACTUAL SE PREDICE A PARTIR DE OTRA IMAGEN DE REFERENCIA
408
OMITIR EL ELEMENTO SINTÁCTICO DE LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO
410
SEÑALAR LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO

FIG. 6A

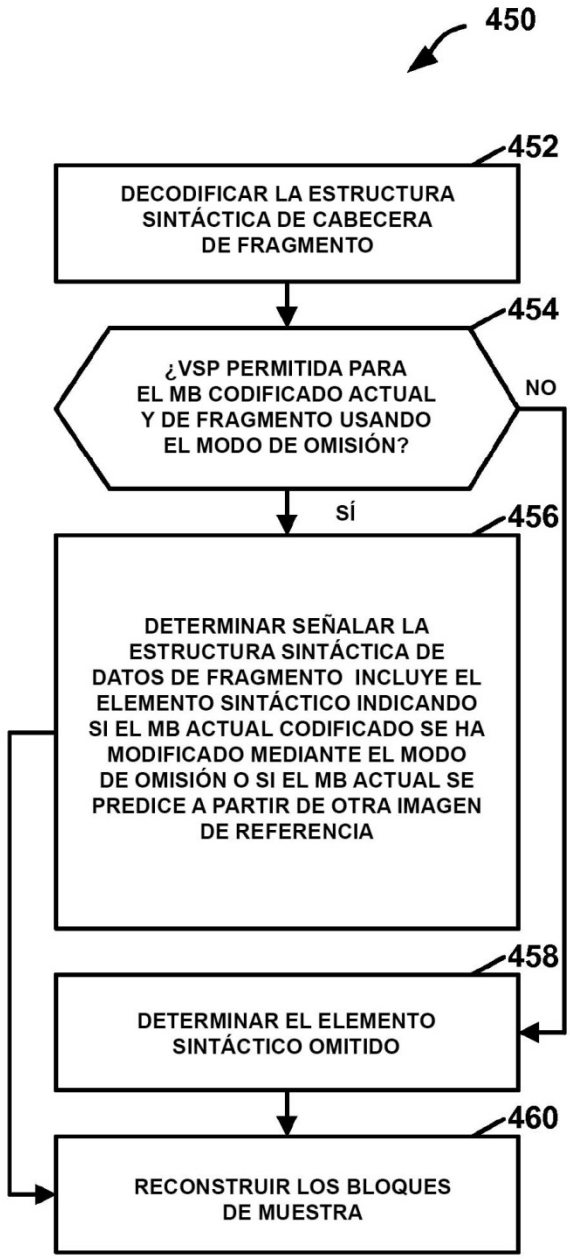
450
452
DECODIFICAR LA ESTRUCTURA SINTÁCTICA DE CABECERA DE FRAGMENTO
454
¿VSP PERMITIDA PARA EL MB CODIFICADO ACTUAL Y DE FRAGMENTO USANDO EL MODO DE OMISIÓN?
NO
SÍ
456
DETERMINAR SEÑALAR LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO INCLUYE EL ELEMENTO SINTÁCTICO INDICANDO SI EL MB ACTUAL CODIFICADO SE HA MODIFICADO MEDIANTE EL MODO DE OMISIÓN O SI EL MB ACTUAL SE PREDICE A PARTIR DE OTRA IMAGEN DE REFERENCIA
458
DETERMINAR EL ELEMENTO SINTÁCTICO OMITIDO
460
RECONSTRUIR LOS BLOQUES DE MUESTRA

FIG. 6B

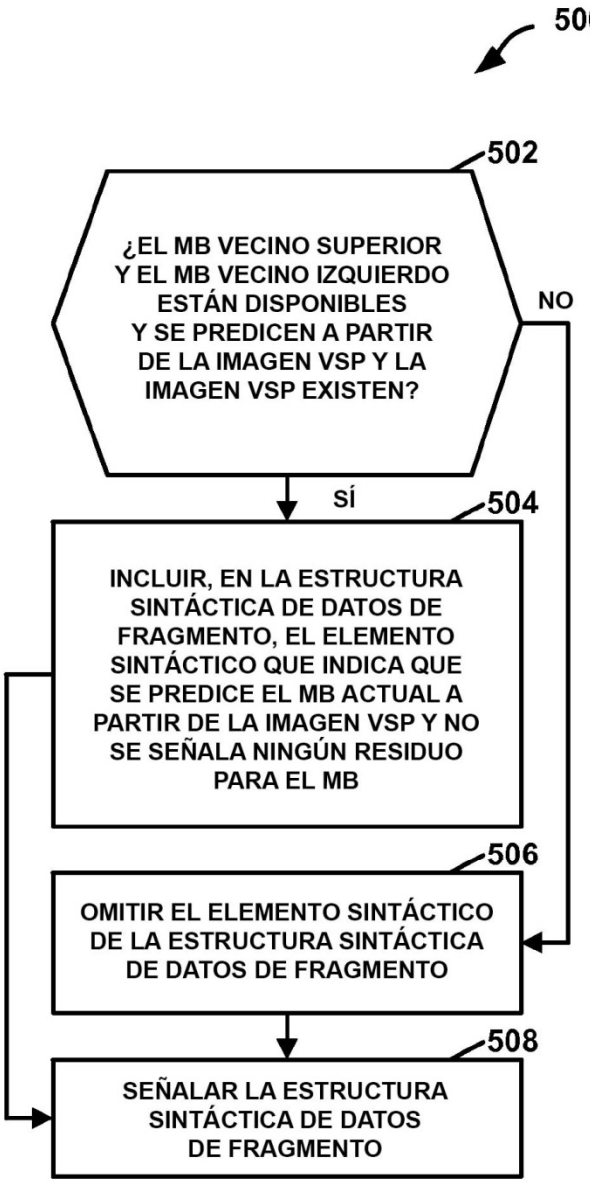
500
502
¿EL MB VECINO SUPERIOR Y EL MB VECINO IZQUIERDO ESTÁN DISPONIBLES Y SE PREDICEN A PARTIR DE LA IMAGEN VSP Y LA IMAGEN VSP EXISTEN?
NO
SÍ
504
INCLUIR, EN LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO, EL ELEMENTO SINTÁCTICO QUE INDICA QUE SE PREDICE EL MB ACTUAL A PARTIR DE LA IMAGEN VSP Y NO SE SEÑALA NINGÚN RESIDUO PARA EL MB
506
OMITIR EL ELEMENTO SINTÁCTICO DE LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO
508
SEÑALAR LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO

FIG. 7A

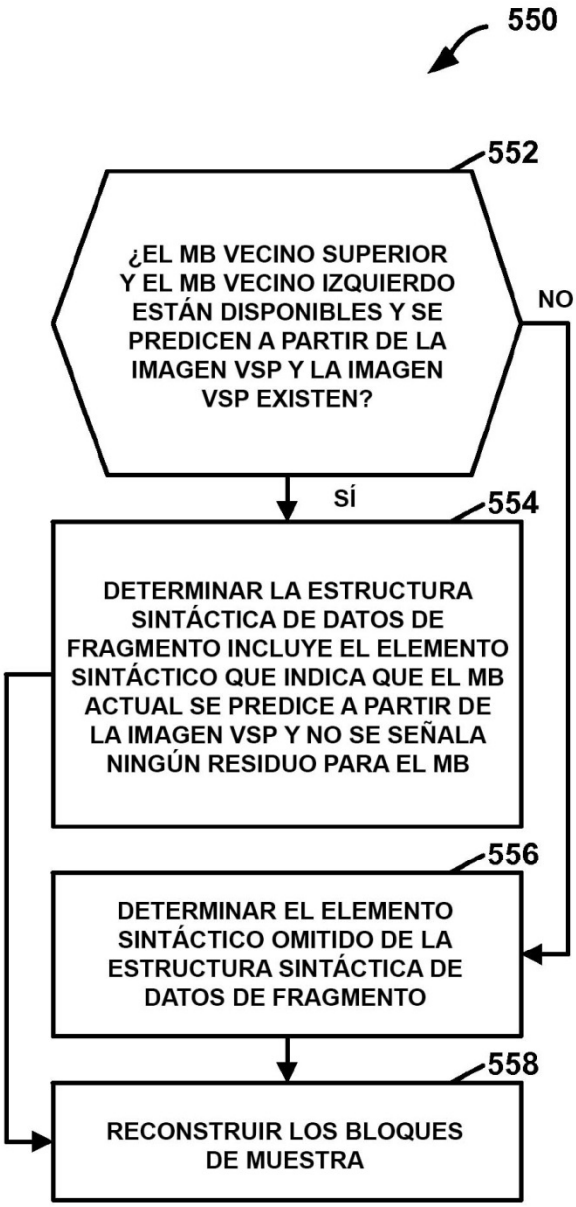
550
552
¿EL MB VECINO SUPERIOR Y EL MB VECINO IZQUIERDO ESTÁN DISPONIBLES Y SE PREDICEN A PARTIR DE LA IMAGEN VSP Y LA IMAGEN VSP EXISTEN?
NO
SÍ
554
DETERMINAR LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO INCLUYE EL ELEMENTO SINTÁCTICO QUE INDICA QUE EL MB ACTUAL SE PREDICE A PARTIR DE LA IMAGEN VSP Y NO SE SEÑALA NINGÚN RESIDUO PARA EL MB
556
DETERMINAR EL ELEMENTO SINTÁCTICO OMITIDO DE LA ESTRUCTURA SINTÁCTICA DE DATOS DE FRAGMENTO
558
RECONSTRUIR LOS BLOQUES DE MUESTRA

FIG. 7B

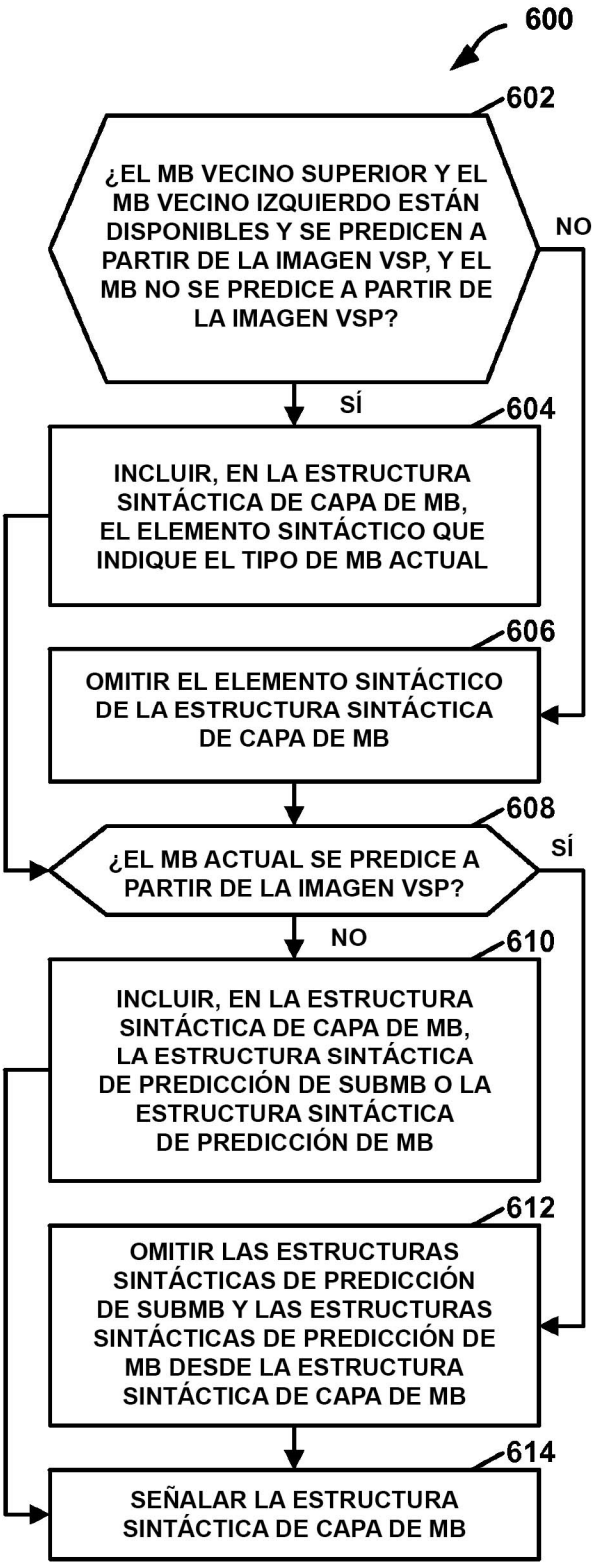
600
602
¿EL MB VECINO SUPERIOR Y EL MB VECINO IZQUIERDO ESTÁN DISPONIBLES Y SE PREDICEN A PARTIR DE LA IMAGEN VSP, Y EL MB NO SE PREDICE A PARTIR DE LA IMAGEN VSP?
NO
SÍ
604
INCLUIR, EN LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB, EL ELEMENTO SINTÁCTICO QUE INDIQUE EL TIPO DE MB ACTUAL
606
OMITIR EL ELEMENTO SINTÁCTICO DE LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB
608
¿EL MB ACTUAL SE PREDICE A PARTIR DE LA IMAGEN VSP?
SÍ
NO
610
INCLUIR, EN LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB, LA ESTRUCTURA SINTÁCTICA DE PREDICCIÓN DE SUBMB O LA ESTRUCTURA SINTÁCTICA DE PREDICCIÓN DE MB
612
OMITIR LAS ESTRUCTURAS SINTÁCTICAS DE PREDICCIÓN DE SUBMB Y LAS ESTRUCTURAS SINTÁCTICAS DE PREDICCIÓN DE MB DESDE LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB
614
SEÑALAR LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB

FIG. 8A

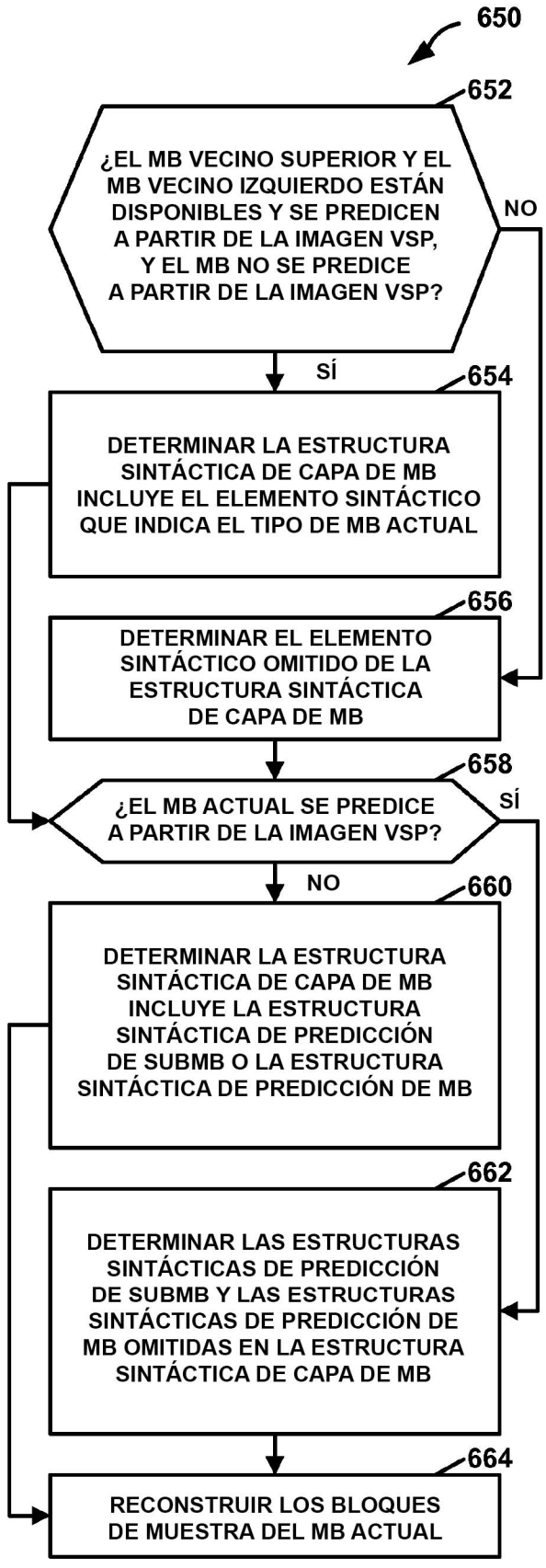
650
652
¿EL MB VECINO SUPERIOR Y EL MB VECINO IZQUIERDO ESTÁN DISPONIBLES Y SE PREDICEN A PARTIR DE LA IMAGEN VSP, Y EL MB NO SE PREDICE A PARTIR DE LA IMAGEN VSP?
NO
SÍ
654
DETERMINAR LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB INCLUYE EL ELEMENTO SINTÁCTICO QUE INDICA EL TIPO DE MB ACTUAL
656
DETERMINAR EL ELEMENTO SINTÁCTICO OMITIDO DE LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB
658
¿EL MB ACTUAL SE PREDICE A PARTIR DE LA IMAGEN VSP?
SÍ
NO
660
DETERMINAR LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB INCLUYE LA ESTRUCTURA SINTÁCTICA DE PREDICCIÓN DE SUBMB O LA ESTRUCTURA SINTÁCTICA DE PREDICCIÓN DE MB
662
DETERMINAR LAS ESTRUCTURAS SINTÁCTICAS DE PREDICCIÓN DE SUBMB Y LAS ESTRUCTURAS SINTÁCTICAS DE PREDICCIÓN DE MB OMITIDAS EN LA ESTRUCTURA SINTÁCTICA DE CAPA DE MB
664
RECONSTRUIR LOS BLOQUES DE MUESTRA DEL MB ACTUAL

FIG. 8B

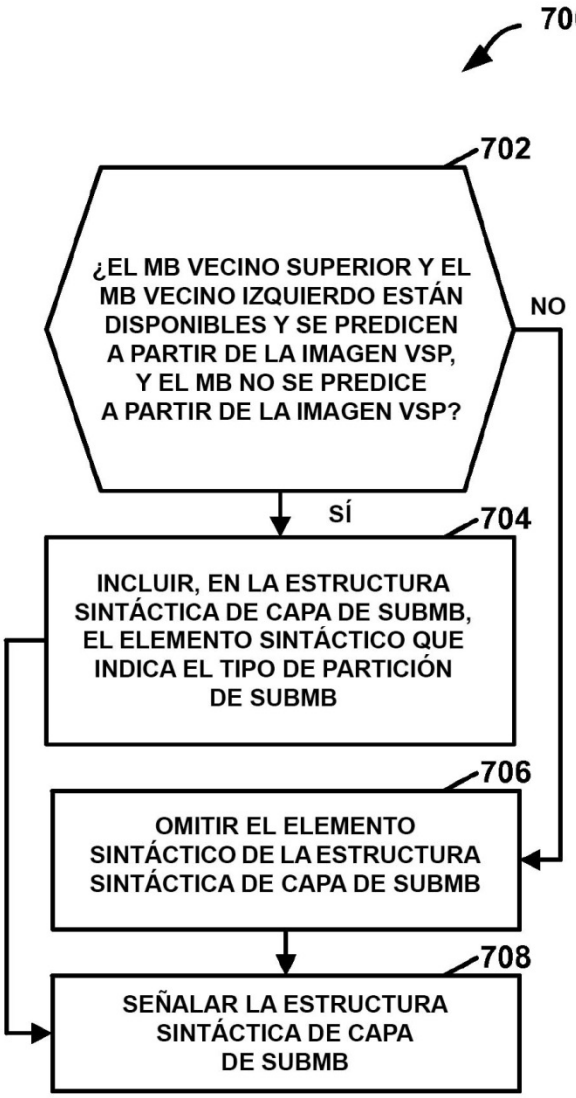
700
702
¿EL MB VECINO SUPERIOR Y EL MB VECINO IZQUIERDO ESTÁN DISPONIBLES Y SE PREDICEN A PARTIR DE LA IMAGEN VSP, Y EL MB NO SE PREDICE A PARTIR DE LA IMAGEN VSP?
NO
SÍ
704
INCLUIR, EN LA ESTRUCTURA SINTÁCTICA DE CAPA DE SUBMB, EL ELEMENTO SINTÁCTICO QUE INDICA EL TIPO DE PARTICIÓN DE SUBMB
706
OMITIR EL ELEMENTO SINTÁCTICO DE LA ESTRUCTURA SINTÁCTICA DE CAPA DE SUBMB
708
SEÑALAR LA ESTRUCTURA SINTÁCTICA DE CAPA DE SUBMB

FIG. 9A

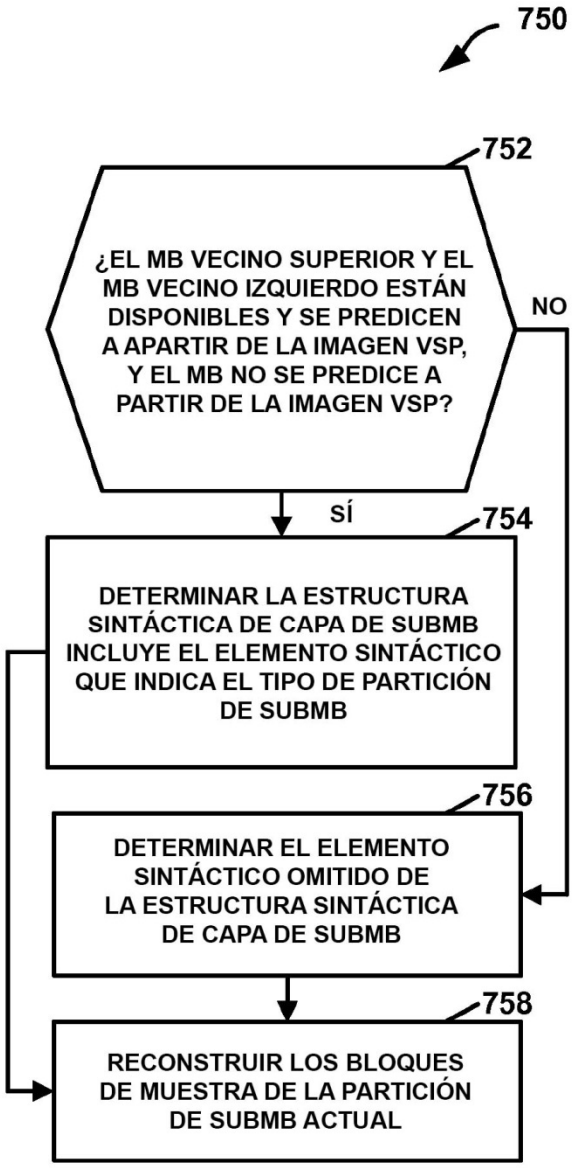
750
752
¿EL MB VECINO SUPERIOR Y EL MB VECINO IZQUIERDO ESTÁN DISPONIBLES Y SE PREDICEN A APARTIR DE LA IMAGEN VSP, Y EL MB NO SE PREDICE A PARTIR DE LA IMAGEN VSP?
NO
SÍ
754
DETERMINAR LA ESTRUCTURA SINTÁCTICA DE CAPA DE SUBMB INCLUYE EL ELEMENTO SINTÁCTICO QUE INDICA EL TIPO DE PARTICIÓN DE SUBMB
756
DETERMINAR EL ELEMENTO SINTÁCTICO OMITIDO DE LA ESTRUCTURA SINTÁCTICA DE CAPA DE SUBMB
758
RECONSTRUIR LOS BLOQUES DE MUESTRA DE LA PARTICIÓN DE SUBMB ACTUAL

FIG. 9B

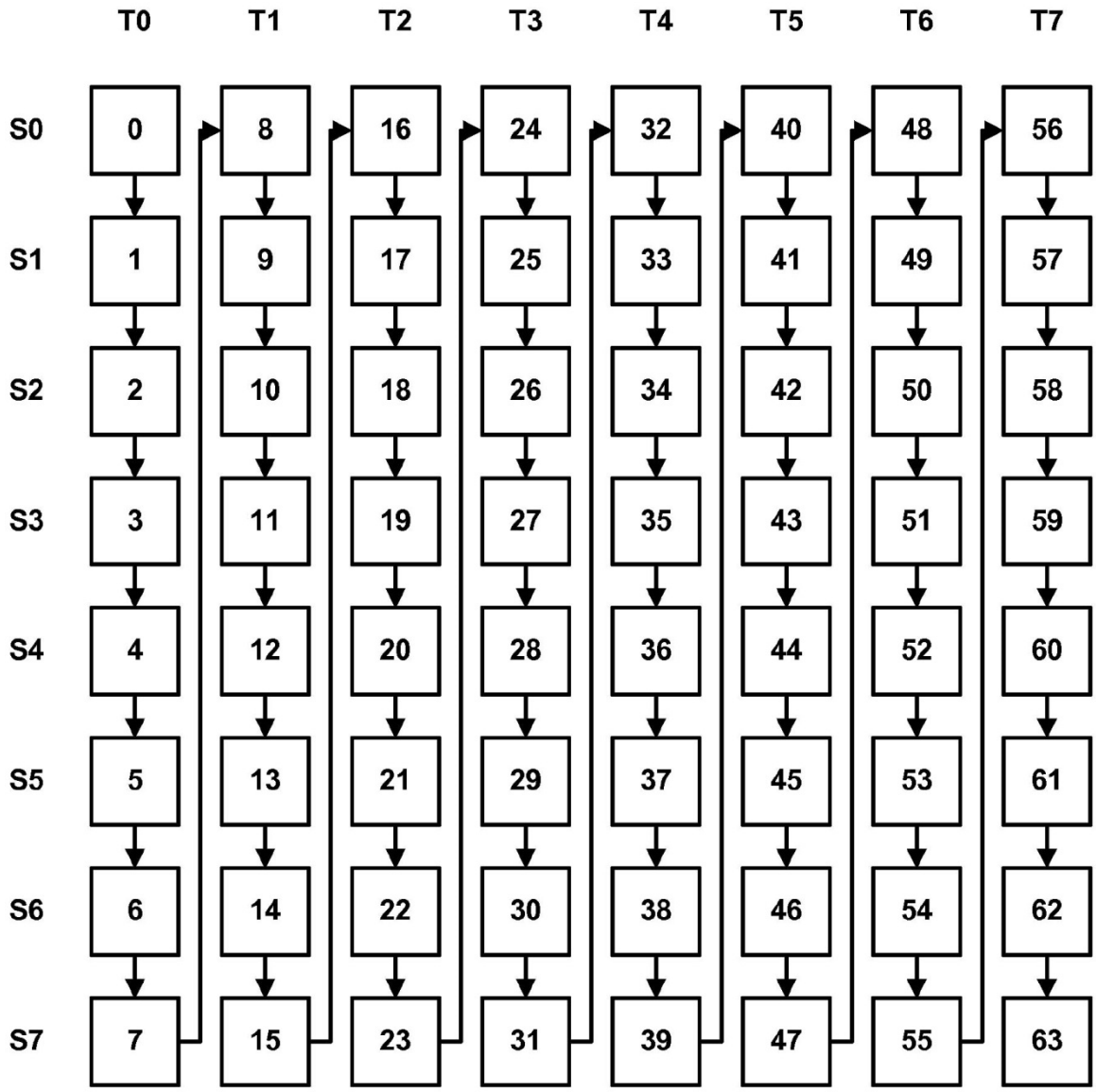
T0
T1
T2
T3
T4
T5
T6
T7
S0
S1
S2
S3
S4
S5
S6
S7
0
8
16
24
32
40
48
56
1
9
17
25
33
41
49
57
2
10
18
26
34
42
50
58
3
11
19
27
35
43
51
59
4
12
20
28
36
44
52
60
5
13
21
29
37
45
53
61
6
14
22
30
38
46
54
62
7
15
23
31
39
47
55
63

FIG. 10

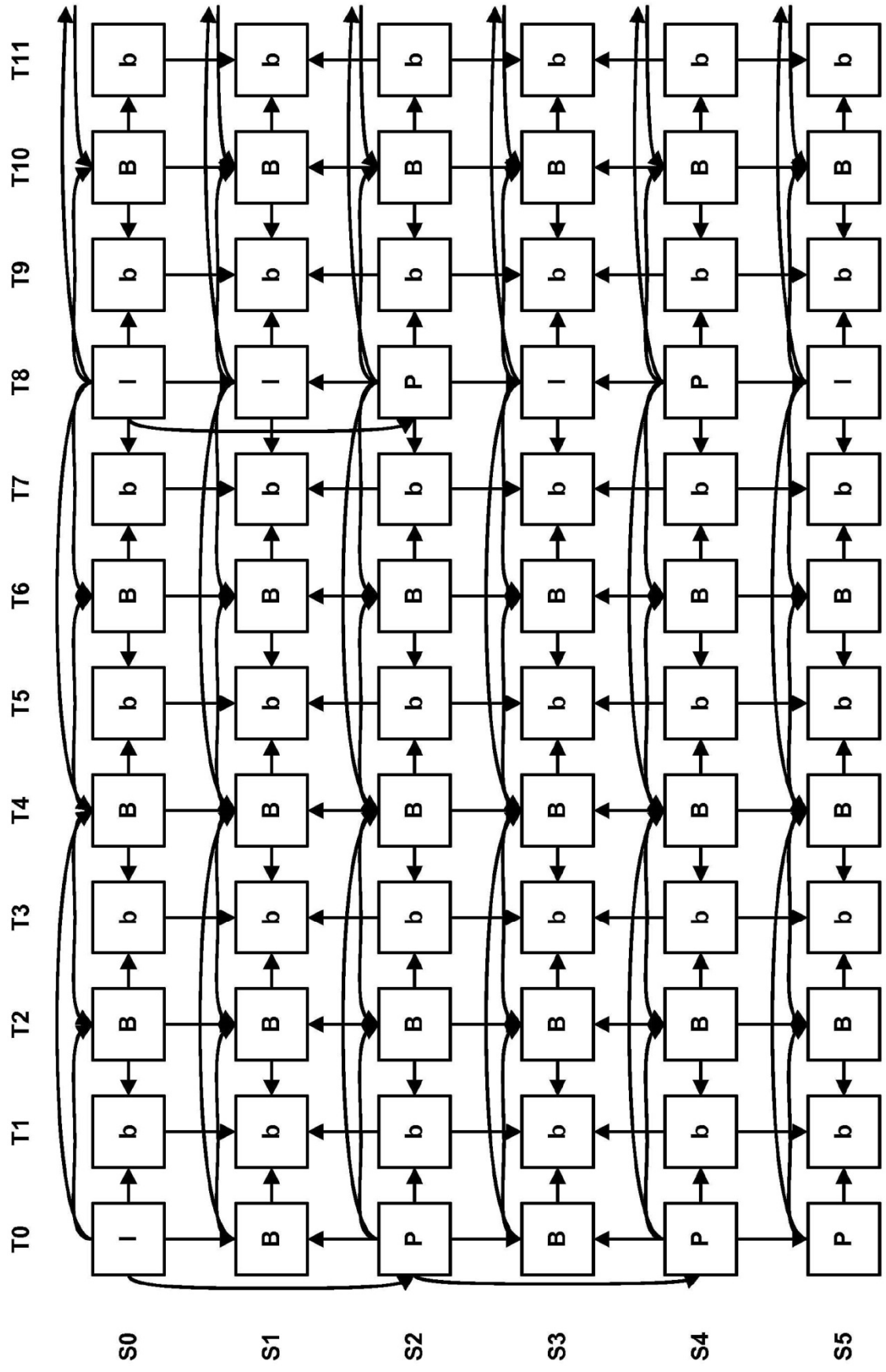

FIG. 11