

19



OFICINA ESPAÑOLA DE  
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **3 024 137**

51 Int. Cl.:

**H04N 19/70** (2014.01)

**H04N 19/13** (2014.01)

**H04N 19/36** (2014.01)

**H04N 19/30** (2014.01)

12

## TRADUCCIÓN DE PATENTE EUROPEA

T3

86 Fecha de presentación y número de la solicitud internacional: **13.09.2019** **PCT/US2019/051149**

87 Fecha y número de publicación internacional: **19.03.2020** **WO20056354**

96 Fecha de presentación y número de la solicitud europea: **13.09.2019** **E 19859301 (4)**

97 Fecha y número de publicación de la concesión europea: **26.02.2025** **EP 3850852**

54 Título: **Direccionamiento basado en mosaicos en codificación de vídeo**

30 Prioridad:

**14.09.2018 US 201862731696 P**

45 Fecha de publicación y mención en BOPI de la traducción de la patente:  
**03.06.2025**

73 Titular/es:

**HUAWEI TECHNOLOGIES CO., LTD. (100.00%)**  
**Huawei Administration Building, Bantian,**  
**Longgang District**  
**Shenzhen, Guangdong 518129, CN**

72 Inventor/es:

**WANG, YE-KUI;**  
**HENDRY, FNU y**  
**CHEN, JIANLE**

74 Agente/Representante:

**PONS ARIÑO, Ángel**

ES 3 024 137 T3

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

## DESCRIPCIÓN

Direccionamiento basado en mosaicos en codificación de vídeo

### 5 Campo técnico

La presente descripción está relacionada generalmente con la codificación de vídeo, y está relacionada específicamente con el particionamiento de imágenes en segmentos, mosaicos y unidades de árbol de codificación (CTU) para soportar una mayor compresión en la codificación de vídeo.

10

### Antecedentes

La cantidad de datos de vídeo necesarios para representar incluso un vídeo relativamente corto puede ser sustancial, lo que puede dar lugar a dificultades cuando los datos se van a transmitir o comunicar de otro modo en una red de comunicaciones con capacidad de ancho de banda limitada. Por lo tanto, los datos de vídeo generalmente se comprimen antes de comunicarse en las redes de telecomunicaciones modernas. El tamaño de un vídeo también podría ser un problema cuando el vídeo se almacena en un dispositivo de almacenamiento debido a que los recursos de memoria pueden ser limitados. Los dispositivos de compresión de vídeo a menudo usan software y/o hardware en el origen para codificar los datos de vídeo antes de la transmisión o el almacenamiento, lo que reduce la cantidad de datos necesarios para representar imágenes de vídeo digital. Luego, los datos comprimidos son recibidos en el destino por un dispositivo de descompresión de vídeo que decodifica los datos de vídeo. Con recursos de red limitados y demandas cada vez mayores de mayor calidad de vídeo, son deseables técnicas mejoradas de compresión y descompresión que mejoren la relación de compresión con poco o ningún sacrificio en la calidad de imagen.

25

El documento EP 2 772 055 A1 describe un método de codificación de datos de vídeo que comprende una imagen que está particionada en una pluralidad de mosaicos.

### Compendio

30

La invención se describe en el conjunto de reivindicaciones adjuntas. Para lograr el objetivo anterior, se adoptan las soluciones técnicas definidas en las reivindicaciones adjuntas. La invención se define en las reivindicaciones independientes. En las reivindicaciones dependientes se proporcionan características adicionales de la invención. A continuación, las partes de la descripción y los dibujos que se refieren a realizaciones que no están cubiertas por las reivindicaciones no se presentan como realizaciones de la invención, sino como ejemplos útiles para comprender la invención.

35

### Breve descripción de los dibujos

Para una comprensión más completa de esta descripción, se hace referencia ahora a la siguiente descripción breve, tomada en relación con los dibujos adjuntos y la descripción detallada, en donde los números de referencia iguales representan partes iguales.

40

La FIG. 1 es un diagrama de flujo de un método de ejemplo de codificación de una señal de vídeo.

45

La FIG. 2 es un diagrama esquemático de un sistema de codificación y decodificación (códec) de ejemplo para codificación de vídeo.

La FIG. 3 es un diagrama esquemático que ilustra un codificador de vídeo de ejemplo.

50

La FIG. 4 es un diagrama esquemático que ilustra un decodificador de vídeo de ejemplo.

La FIG. 5 es un diagrama esquemático que ilustra un flujo de bits de ejemplo que contiene una secuencia de vídeo codificada.

55

La FIG. 6 es un diagrama esquemático que ilustra una imagen de ejemplo particionada para codificación.

La FIG. 7 es un diagrama esquemático que ilustra una imagen de ejemplo con límites de segmento definidos por la posición del mosaico.

60

La FIG. 8 es un diagrama esquemático de un dispositivo de codificación de vídeo de ejemplo.

La FIG. 9 es un diagrama de flujo de un método de ejemplo de codificación de una imagen en un flujo de bits basándose en límites de segmento definidos por la posición del mosaico.

65

La FIG. 10 es un diagrama de flujo de un método de ejemplo de decodificación de una imagen de un flujo de

bits basándose en límites de segmento definidos por la posición del mosaico.

La FIG. 11 es un diagrama esquemático de un sistema de ejemplo para codificar una secuencia de vídeo de imágenes en un flujo de bits basándose en límites de segmento definidos por la posición del mosaico.

5

#### Descripción detallada

Se pueden emplear muchas técnicas de compresión de vídeo para reducir el tamaño de los archivos de vídeo con una pérdida mínima de datos. Por ejemplo, las técnicas de compresión de vídeo pueden incluir realización de predicción espacial (por ejemplo, intraimagen) y/o predicción temporal (por ejemplo, interimagen) para reducir o eliminar redundancia de datos en secuencias de vídeo. Para la codificación de vídeo basada en bloques, un segmento de vídeo (es decir, una imagen de vídeo o una parte de una imagen de vídeo) puede partitionarse en bloques de vídeo, que también pueden denominarse bloques de árbol, bloques de árbol de codificación (CTB), unidades de árbol de codificación (CTU), unidades de codificación (CU) y/o nodos de codificación. Los bloques de vídeo en un segmento intracodificado (I) de una imagen se codifican usando predicción espacial con respecto a muestras de referencia en bloques vecinos en la misma imagen. Los bloques de vídeo en un segmento de predicción unidireccional (P) o bidireccional (B) de una imagen pueden codificarse empleando predicción espacial con respecto a muestras de referencia en bloques vecinos en la misma imagen o predicción temporal con respecto a muestras de referencia en otras imágenes de referencia. Las imágenes pueden denominarse fotogramas y/o imágenes, y las imágenes de referencia pueden denominarse fotogramas de referencia y/o imágenes de referencia. La predicción espacial o temporal da como resultado un bloque predictivo que representa un bloque de imagen. Los datos residuales representan diferencias de píxeles entre el bloque original de imagen y el bloque predictivo. En consecuencia, un bloque intercodificado se codifica según un vector de movimiento que apunta a un bloque de muestras de referencia que forman el bloque predictivo, y los datos residuales que indican la diferencia entre el bloque codificado y el bloque predictivo. Un bloque intracodificado se codifica según un modo de intracodificación y los datos residuales. Para una mayor compresión, los datos residuales se transforman del dominio de píxeles a un dominio de transformación. Estos dan como resultado coeficientes de transformación residuales, que pueden cuantificarse. Los coeficientes de transformación cuantificados pueden organizarse inicialmente en una matriz bidimensional. Los coeficientes de transformación cuantificados se pueden escanear para producir un vector unidimensional de coeficientes de transformación. Se puede aplicar codificación de entropía para lograr adicionalmente compresión. Estas técnicas de compresión de vídeo se analizan con mayor detalle a continuación.

Para garantizar que un vídeo codificado pueda decodificarse con precisión, el vídeo se codifica y decodifica según los estándares de codificación de vídeo correspondientes. Tales estándares de codificación de vídeo incluyen H.261 del Sector de Estandarización de la Unión Internacional de Telecomunicaciones (ITU) (ITU-T), (MPEG)-1 Parte 2 del Grupo de Expertos de Imágenes en Movimiento de la Organización Internacional de Estandarización/Comisión Electrotécnica Internacional (ISO/IEC), H.262 de ITU-T o Parte 2 de MPEG-2 de ISO/IEC, H.263 de ITU-T, Partes 2 de MPEG-4 de ISO/IEC, Codificación de Vídeo Avanzada (AVC), también conocida como H.264 de ITU-T o Parte 10 de MPEG-4 de ISO/IEC, y Codificación de Vídeo de Alta Eficiencia (HEVC), también conocida como H.265 de ITU-T o Parte 2 de MPEG-H. La AVC incluye extensiones tales como Codificación de Vídeo Escalable (SVC), Codificación de Vídeo Multivista (MVC) y Codificación de Vídeo Multivista más Profundidad (MVC+D) y AVC tridimensional (3D) (3D-AVC). La HEVC incluye extensiones tales como HEVC Escalable (SHVC), HEVC Multivista (MV-HEVC) y HEVC 3D (3D-HEVC). El equipo conjunto de expertos en vídeo (JVET) del ITU-T y la ISO/IEC ha empezado a desarrollar un estándar de codificación de vídeo denominado Codificación de Vídeo Versátil (VVC). La VVC está incluida en un Borrador de Trabajo (WD), que incluye JVET-K1001-v4 y JVET-K1002-v1.

Para codificar una imagen de vídeo, primero se particiona la imagen y las particiones se codifican en un flujo de bits. Hay varios esquemas de partición de imágenes disponibles. Por ejemplo, una imagen se puede particionar en segmentos regulares, segmentos dependientes, mosaicos y/o según el Procesamiento Paralelo de Frente de Onda (WPP). Para simplificar, la HEVC restringe los codificadores para que solo se puedan usar segmentos regulares, segmentos dependientes, mosaicos, WPP y combinaciones de los mismos al particionar un segmento en grupos de CTB para codificación de vídeo. Esta partición se puede aplicar para soportar la coincidencia del tamaño de la Unidad de Transferencia Máxima (MTU), el procesamiento paralelo y la reducción de retardo de extremo a extremo. La MTU denota la cantidad máxima de datos que se pueden transmitir en un solo paquete. Si la carga útil de paquete excede la MTU, esa carga útil se divide en dos paquetes a través de un proceso llamado fragmentación.

60

Un segmento regular, también denominado simplemente como segmento, es un segmento particionado de una imagen que se puede reconstruir independientemente de otros segmentos regulares dentro de la misma imagen, a pesar de algunas interdependencias debidas a operaciones de filtrado de bucle. Cada segmento regular se encapsula en su propia unidad de Capa de Abstracción de Red (NAL) para su transmisión. Además, la predicción en imagen (predicción intramuestra, predicción de información de movimiento, predicción del modo de codificación) y la dependencia de la codificación de entropía en los límites de los segmentos se

65

pueden deshabilitar para soportar la reconstrucción independiente. Tal reconstrucción independiente soporta la paralelización. Por ejemplo, la paralelización regular basada en segmento regular emplea una comunicación mínima entre procesadores o entre núcleos. Sin embargo, como cada segmento regular es independiente, cada segmento está asociado con una cabecera de segmento separada. El uso de segmentos regulares puede incurrir en una sobrecarga de codificación sustancial debido al coste en bits de la cabecera de segmento para cada segmento y debido a la falta de predicción en los límites del segmento. Además, se pueden emplear segmentos regulares para soportar la coincidencia con los requisitos de tamaño para la MTU. Específicamente, como un segmento regular se encapsula en una unidad NAL separada y se puede codificar independientemente, cada segmento regular debe ser más pequeño que la MTU en los esquemas MTU para evitar dividir el segmento en múltiples paquetes. Como tal, el objetivo de paralelización y el objetivo de coincidencia del tamaño de MTU pueden plantear demandas contradictorias para el diseño de un segmento en una imagen.

Los segmentos dependientes son similares a los segmentos regulares, pero tienen cabeceras de segmento más cortas y permiten el particionamiento de los límites del bloque del árbol de imágenes sin interrumpir la predicción en imagen. En consecuencia, los segmentos dependientes permiten que un segmento regular se fragmente en múltiples unidades NAL, lo que proporciona un retardo de extremo a extremo reducido al permitir que se envíe una parte de un segmento regular antes de que se complete la codificación de todo el segmento regular.

Un mosaico es un segmento particionado de una imagen creado por límites horizontales y verticales que crean columnas y filas de mosaicos. Los mosaicos se pueden codificar en orden de escaneo de trama (de derecha a izquierda y de arriba a abajo). El orden de escaneo de los CTB es local dentro de un mosaico. En consecuencia, los CTB en un primer mosaico se codifican en orden de escaneo de trama, antes de proceder a los CTB en el siguiente mosaico. De manera similar a los segmentos regulares, los mosaicos rompen las dependencias de predicción en imagen, así como las dependencias de decodificación de entropía. Sin embargo, es posible que los mosaicos no se incluyan en unidades NAL individuales y, por lo tanto, es posible que no se usen para la coincidencia del tamaño de MTU. Cada mosaico puede ser procesado por un procesador/núcleo, y la comunicación entre procesadores/entre núcleos empleada para la predicción en imagen entre unidades de procesamiento que decodifican mosaicos vecinos puede limitarse a transmitir una cabecera de segmento compartida (cuando los mosaicos adyacentes están en el mismo segmento) y realizar el filtrado de bucle relacionado con el intercambio de muestras reconstruidas y metadatos. Cuando se incluye más de un mosaico en un segmento, el desplazamiento de bytes del punto de entrada para cada mosaico que no sea el primer desplazamiento del punto de entrada en el segmento se puede señalar en la cabecera de segmento. Para cada segmento y mosaico, se debe cumplir al menos una de las siguientes condiciones: 1) todos los bloques de árbol codificados en un segmento pertenecen al mismo mosaico; y 2) todos los bloques de árbol codificados en un mosaico pertenecen al mismo segmento.

En WPP, la imagen se particiona en filas individuales de CTB. Los mecanismos de predicción y decodificación de entropía pueden usar datos de CTB en otras filas. El procesamiento paralelo es posible gracias a la decodificación paralela de filas CTB. Por ejemplo, una fila actual puede decodificarse en paralelo con una fila anterior. Sin embargo, la decodificación de la fila actual se retarda respecto del proceso de decodificación de las filas anteriores por dos CTB. Este retardo garantiza que los datos relacionados con el CTB anterior y el CTB actual y a la derecha del CTB actual en la fila actual estén disponibles antes de que se codifique el CTB actual. Este enfoque aparece como un frente de onda cuando se representa gráficamente. Este inicio escalonado permite la paralelización con hasta tantos procesadores/núcleos como filas CTB contenga la imagen. Debido a que se permite la predicción en imagen entre filas de bloques de árbol vecinas dentro de una imagen, la comunicación entre procesadores y núcleos para habilitar la predicción en imagen puede ser sustancial. El particionamiento WPP considera los tamaños de unidad NAL. Por lo tanto, WPP no soporta la coincidencia del tamaño de MTU. Sin embargo, se pueden usar segmentos regulares junto con WPP, con cierta sobrecarga de codificación, para implementar la coincidencia del tamaño de MTU según se desee.

Los mosaicos también pueden incluir conjuntos de mosaicos con restricciones de movimiento. Un Conjunto de Mosaicos con Restricciones de Movimiento (MCTS) es un conjunto de mosaicos diseñado de manera que los vectores de movimiento asociados están restringidos para apuntar a ubicaciones de muestra completa dentro del MCTS y a ubicaciones de muestra fraccionaria que requieren solo ubicaciones de muestra completa dentro del MCTS para la interpolación. Además, no se permite el uso de candidatos de vectores de movimiento para la predicción de vectores de movimiento temporal derivados de bloques fuera del MCTS. De esta manera, cada MCTS puede ser decodificado independientemente sin la existencia de mosaicos no incluidos en el MCTS. Los mensajes de información de mejora complementaria (SEI) de MCTS temporales se pueden usar para indicar la existencia de MCTS en el flujo de bits y señalar los MCTS. El mensaje SEI de MCTS proporciona información complementaria que se puede usar en la extracción del subflujo de bits de MCTS (especificado como parte de la semántica del mensaje SEI) para generar un flujo de bits conforme a un MCTS. La información incluye un número de conjuntos de información de extracción, cada uno de los cuales define un número de MCTS y contiene bytes de carga útil de secuencia de bytes sin procesar (RBSP) de los conjuntos de parámetros de vídeo de reemplazo (VPS), conjuntos de parámetros de secuencia (SPS) y conjuntos de parámetros de

imagen (PPS) que se usarán durante el proceso de extracción de subflujo de bits MCTS. Al extraer un subflujo de bits según el proceso de extracción de subflujo de bits MCTS, los conjuntos de parámetros (VPS, SPS y PPS) se pueden reescribir o reemplazar, y las cabeceras de segmento se pueden actualizar porque uno o todos los elementos de sintaxis relacionados con la dirección de segmento (incluidos first\_slice\_segment\_in\_pic\_flag y slice\_segment\_address) pueden emplear valores diferentes en el subflujo de bits extraído.

Los mecanismos de segmentación y división anteriores proporcionan una flexibilidad significativa para soportar la coincidencia del tamaño de MTU y el procesamiento paralelo. Sin embargo, la coincidencia del tamaño de MTU se ha vuelto menos relevante debido a la creciente velocidad y fiabilidad de las redes de telecomunicaciones. Por ejemplo, uno de los usos principales de la coincidencia del tamaño de MTU es soportar la visualización de imágenes con errores ocultos. Una imagen con errores ocultos es una imagen decodificada que se crea a partir de una imagen codificada transmitida cuando hay alguna pérdida de datos. Tal pérdida de datos puede incluir una pérdida de algunos segmentos de una imagen codificada o errores en las imágenes de referencia usadas por la imagen codificada (por ejemplo, la imagen de referencia también es una imagen con errores ocultos). Se puede crear una imagen con errores ocultos visualizando los segmentos correctos y estimando los segmentos erróneos, por ejemplo, copiando un segmento correspondiente al segmento erróneo de la imagen anterior en la secuencia de vídeo. Se pueden generar imágenes con errores ocultos cuando cada segmento está contenido en una sola unidad NAL. Sin embargo, si los segmentos están fragmentados en múltiples unidades NAL (por ejemplo, no hay coincidencia del tamaño de MTU), la pérdida de una unidad NAL puede dañar múltiples segmentos. La generación de imágenes con errores ocultos es menos relevante en los entornos de red modernos, ya que la pérdida de paquetes es un fenómeno mucho menos común y porque las velocidades de red modernas permiten que el sistema omita por completo las imágenes con errores sin causar una congelación significativa del vídeo, ya que el retardo entre una imagen errónea y una imagen siguiente sin errores es generalmente pequeño. Además, el proceso para estimar la calidad de una imagen con errores ocultos puede ser complicado y, por lo tanto, puede ser preferible simplemente omitir la imagen errónea. En consecuencia, las aplicaciones de conversación por vídeo, tales como las videoconferencia y la videotelefonía, e incluso las aplicaciones de transmisión, generalmente renuncian a usar imágenes con errores ocultos.

Como las imágenes con errores ocultos son menos útiles, la coincidencia del tamaño de MTU es menos útil. Además, continuar soportando paradigmas de coincidencia del tamaño de MTU durante el particionamiento puede complicar innecesariamente los sistemas de codificación y también usar bits residuales que de otro modo podrían omitirse para aumentar la eficiencia de la codificación. Además, algunos esquemas de mosaico (por ejemplo, MCTS) permiten visualizar subimágenes de una imagen. Para visualizar una subimagen, se visualizan segmentos en una región de interés y se omiten otros segmentos. La región de interés puede comenzar en una ubicación distinta a la parte superior izquierda de la imagen y, por lo tanto, puede tener direcciones que estén desplazadas desde el inicio de la imagen por un valor variable. Para visualizar correctamente la subimagen, se puede usar un empalmador para reescribir la(s) cabecera(s) de segmento para la región de interés para tener en cuenta este desplazamiento. Sería beneficioso contar con un esquema de segmentación y agrupación en mosaicos que no requiriera la reescritura de las cabeceras de segmento. Además, los límites de mosaico no pueden tratarse como límites de imagen a menos que estén cubiertos junto a los límites de imagen. Sin embargo, tratar los límites de mosaico como límites de imagen puede aumentar la eficiencia de la codificación en algunos casos debido al relleno de los límites y debido a la relajación de las restricciones relacionadas con los vectores de movimiento que apuntan a muestras fuera de los límites en las imágenes de referencia. Además, la HEVC puede emplear un indicador denominado end\_of\_slice\_flag al final de los datos codificados para cada CTU para indicar si se ha alcanzado el final del segmento. La AVC emplea este indicador al final de los datos codificados para cada macrobloque (MB) para el mismo propósito. Sin embargo, la codificación de este indicador es innecesaria y un desperdicio de bits cuando la última CTU/MB se conoce a través de otros mecanismos. La presente descripción presenta mecanismos para abordar estos y otros problemas en las técnicas de codificación de vídeo.

Se describen en la presente memoria varios mecanismos para aumentar la eficiencia de codificación y reducir la sobrecarga de procesamiento asociada con los esquemas de segmentación y agrupación en mosaicos analizados anteriormente. En un ejemplo, se requiere que los segmentos incluyan un número entero de mosaicos. Además, es necesario que los segmentos sean rectangulares (incluidos los cuadrados), pero no es necesario que se extiendan por todo el ancho de una imagen. Además, a los mosaicos se les pueden asignar identificadores de mosaico (ID) que son únicos dentro de la imagen y aumentan en orden de trama. Esto permite que los límites de los segmentos se señalicen basándose en el ID de mosaico y no basándose en la posición relativa dentro de la imagen. Por ejemplo, los límites de un primer segmento se pueden señalar basándose en un ID de mosaico superior izquierdo y un ID de mosaico inferior derecho. Luego, un decodificador puede inferir una lista completa de mosaicos e ID de mosaico en el segmento basándose en el ID de mosaico superior izquierdo e inferior derecho. Esto permite que un único conjunto de datos señalice tanto una lista de ID de mosaico como los límites de los segmentos. Además, los segmentos pueden incluir cada uno una o más CTU. Es posible direccionar cada CTU basándose en el ID de mosaico que contiene las CTU. En algunos ejemplos, las direcciones de las CTU se pueden señalar en el flujo de bits. En otros ejemplos, las direcciones

de las CTU también se pueden inferir en el decodificador y, por lo tanto, omitirse del flujo de bits para aumentar la eficiencia de la codificación. Estos y otros ejemplos se describen con detalle a continuación.

La FIG. 1 es un diagrama de flujo de un método 100 de operación de ejemplo de codificación de una señal de vídeo. Específicamente, una señal de vídeo se codifica en un codificador. El proceso de codificación comprime la señal de vídeo empleando varios mecanismos para reducir el tamaño del archivo de vídeo. Un tamaño de archivo más pequeño permite que el archivo de vídeo comprimido se transmita hacia un usuario, al tiempo que reduce la sobrecarga de ancho de banda asociada. Luego, el decodificador decodifica el archivo de vídeo comprimido para reconstruir la señal de vídeo original para visualizarla a un usuario final. El proceso de decodificación generalmente es un espejo el proceso de codificación para permitir que el decodificador reconstruya consistentemente la señal de vídeo.

En la etapa 101, la señal de vídeo se introduce en el codificador. Por ejemplo, la señal de vídeo puede ser un archivo de vídeo sin comprimir almacenado en memoria. Como otro ejemplo, el archivo de vídeo puede ser capturado por un dispositivo de captura de vídeo, tal como una cámara de vídeo, y codificado para soportar la transmisión en vivo del vídeo. El archivo de vídeo puede incluir tanto un componente de audio como un componente de vídeo. El componente de vídeo contiene una serie de fotogramas de imagen que, cuando se ven en una secuencia, dan la impresión visual de movimiento. Los fotogramas contienen píxeles que se expresan en términos de luz, denominados en la presente memoria componentes de luma (o muestras de luma), y color, que se denominan componentes de croma (o muestras de color). En algunos ejemplos, los fotogramas también pueden contener valores de profundidad para soportar la visualización tridimensional.

En la etapa 103, el vídeo se particiona en bloques. La partición incluye subdividir los píxeles en cada fotograma en bloques cuadrados y/o rectangulares para su compresión. Por ejemplo, en la Codificación de Vídeo de Alta Eficiencia (HEVC) (también conocida como H.265 y Parte 2 de MPEG-H), el fotograma se puede dividir primero en unidades de árbol de codificación (CTU), que son bloques de un tamaño predefinido (por ejemplo, sesenta y cuatro píxeles por sesenta y cuatro píxeles). Las CTU contienen muestras de luma y croma. Se pueden emplear árboles de codificación para dividir las CTU en bloques y luego subdividir los bloques recursivamente hasta que se logren configuraciones que soporten una codificación adicional. Por ejemplo, los componentes de luma de un fotograma se pueden subdividir hasta que los bloques individuales contengan valores de iluminación relativamente homogéneos. Por ejemplo, los componentes de croma de un fotograma se pueden subdividir hasta que los bloques individuales contengan valores de color relativamente homogéneos. En consecuencia, los mecanismos de particionamiento varían dependiendo del contenido de los fotogramas de vídeo.

En la etapa 105, se emplean varios mecanismos de compresión para comprimir los bloques de imagen particionados en la etapa 103. Por ejemplo, se puede emplear la interpredicción y/o la intrapredicción. La interpredicción está diseñada para aprovechar el hecho de que los objetos de una escena común tienden a aparecer en fotogramas sucesivos. En consecuencia, un bloque que representa un objeto en un fotograma de referencia no necesita describirse repetidamente en fotogramas adyacentes. Específicamente, un objeto, tal como una mesa, puede permanecer en una posición constante en múltiples fotogramas. Por lo tanto, la mesa se describe una vez y los fotogramas adyacentes pueden referirse de vuelta al fotograma de referencia. Se pueden emplear mecanismos de coincidencia de patrones para hacer coincidir objetos en múltiples fotogramas. Además, los objetos en movimiento pueden representarse en múltiples fotogramas, por ejemplo, debido al movimiento del objeto o al movimiento de la cámara. Como un ejemplo particular, el vídeo puede mostrar un automóvil que se mueve en la pantalla en múltiples fotogramas. Se pueden emplear vectores de movimiento para describir tal movimiento. Un vector de movimiento es un vector bidimensional que proporciona un desplazamiento desde las coordenadas de un objeto en un fotograma a las coordenadas del objeto en un fotograma de referencia. Como tal, la interpredicción puede codificar un bloque de imagen en un fotograma actual como un conjunto de vectores de movimiento que indican un desplazamiento de un bloque correspondiente en un fotograma de referencia.

La intrapredicción codifica bloques en un fotograma común. La intrapredicción aprovecha el hecho de que los componentes de luma y de croma tienden a agruparse en un fotograma. Por ejemplo, una mancha de verde en una parte de un árbol tiende a colocarse adyacente a manchas similares de verde. La intrapredicción emplea múltiples modos de predicción direccional (por ejemplo, treinta y tres en HEVC), un modo planar y un modo de corriente directa (DC). Los modos direccionales indican que un bloque actual es similar/igual que las muestras de un bloque vecino en una dirección correspondiente. El modo planar indica que una serie de bloques a lo largo de una fila/columna (por ejemplo, un plano) se pueden interpolar basándose en los bloques vecinos en los bordes de la fila. El modo planar, en efecto, indica una transición suave de luz/color en una fila/columna empleando una pendiente relativamente constante en los valores cambiantes. El modo DC se emplea para el suavizado de límites e indica que un bloque es similar/el mismo que un valor promedio asociado con muestras de todos los bloques vecinos asociados con las direcciones angulares de los modos de predicción direccional. En consecuencia, los bloques intrapredicción pueden representar bloques de imágenes como varios valores de modo de predicción relacional en lugar de los valores reales. Además, los bloques de interpredicción pueden representar bloques de imagen como valores de vector de movimiento en lugar de los valores reales. En

cualquier caso, los bloques de predicción pueden no representar exactamente los bloques de imagen en algunos casos. Cualquier diferencia se almacena en bloques residuales. Pueden aplicarse transformaciones a los bloques residuales para comprimir adicionalmente el archivo.

5 En la etapa 107, pueden aplicarse varias técnicas de filtrado. En la HEVC, los filtros se aplican según un esquema de filtrado en bucle. La predicción basada en bloques analizada anteriormente puede dar como resultado la creación de imágenes en bloques en el decodificador. Además, el esquema de predicción basado en bloques puede codificar un bloque y luego reconstruir el bloque codificado para su uso posterior como un bloque de referencia. El esquema de filtrado en bucle aplica iterativamente filtros de supresión de ruido, filtros de desbloqueo, filtros de bucle adaptativo y filtros de offset adaptativo de muestra (SAO) a los bloques/fotogramas. Estos filtros mitigan tales artefactos de bloqueo para que el archivo codificado pueda reconstruirse con precisión. Además, estos filtros mitigan los artefactos en los bloques de referencia reconstruidos, para que sea menos probable que los artefactos creen artefactos adicionales en los bloques posteriores que se codifican basándose en los bloques de referencia reconstruidos.

15 Una vez que la señal de vídeo ha sido particionada, comprimida y filtrada, los datos resultantes se codifican en un flujo de bits en la etapa 109. El flujo de bits incluye los datos analizados anteriormente, así como cualquier dato de señalización deseado para soportar la reconstrucción adecuada de la señal de vídeo en el decodificador. Por ejemplo, tales datos pueden incluir datos de partición, datos de predicción, bloques residuales y varios indicadores que proporcionan instrucciones de codificación al decodificador. El flujo de bits puede ser almacenado en memoria para su transmisión hacia un decodificador bajo demanda. El flujo de bits también puede ser difundido y/o multidifundido hacia una pluralidad de decodificadores. La creación del flujo de bits es un proceso iterativo. En consecuencia, las etapas 101, 103, 105, 107 y 109 pueden ocurrir de forma continua y/o simultánea a lo largo de muchos fotogramas y bloques. El orden que se muestra en la FIG. 1 se presenta para mayor claridad y facilidad del análisis, y no tiene la intención de limitar el proceso de codificación de vídeo a un orden particular.

El decodificador recibe el flujo de bits y comienza el proceso de decodificación en la etapa 111. Específicamente, el decodificador emplea un esquema de decodificación de entropía para convertir el flujo de bits en los datos de sintaxis y vídeo correspondientes. El decodificador emplea los datos de sintaxis del flujo de bits para determinar las particiones de los fotogramas en la etapa 111. La partición debe coincidir con los resultados de la partición de bloques en la etapa 103. Se describe ahora la codificación/decodificación de entropía tal como se emplea en la etapa 111. El codificador realiza muchas decisiones durante el proceso de compresión, tal como seleccionar esquemas de particionamiento de bloques entre varias decisiones posibles basándose en el posicionamiento espacial de los valores en la(s) imagen o imágenes de entrada. La señalización de las decisiones exactas puede emplear un gran número de bins. Como se usa en la presente memoria, un bin es un valor binario que se trata como variable (por ejemplo, un valor de bits que puede variar dependiendo del contexto). La codificación de entropía permite al codificador descartar cualquier decisión que claramente no sea viable para un caso particular, dejando un conjunto de decisiones permitidas. A cada decisión permitida se le asigna una palabra de código. La longitud de las palabras de código se basa en el número de decisiones permitidas (por ejemplo, un bin para dos decisiones, dos bins para tres o cuatro decisiones, etc.). Luego, el codificador codifica la palabra de código para la decisión seleccionada. Este esquema reduce el tamaño de las palabras de código, ya que estas son tan grandes como se desee para indicar de forma única una selección de un pequeño subconjunto de decisiones permitidas, en lugar de indicar de forma única la selección de un conjunto potencialmente grande de todas las decisiones posibles. El decodificador luego decodifica la selección determinando el conjunto de decisiones permitidas de una manera similar al codificador. Determinando el conjunto de decisiones permitidas, el decodificador puede leer la palabra de código y determinar la selección realizada por el codificador.

50 En la etapa 113, el decodificador realiza la decodificación de bloques. Específicamente, el decodificador emplea transformaciones inversas para generar bloques residuales. Luego, el decodificador emplea los bloques residuales y los bloques de predicción correspondientes para reconstruir los bloques de imagen según el particionamiento. Los bloques de predicción pueden incluir tanto bloques de intrapredicción como bloques de interpredicción tal como se generan en el codificador en la etapa 105. Luego, los bloques de imagen reconstruidos se colocan en fotogramas de una señal de vídeo reconstruida según los datos de partición determinados en la etapa 111. La sintaxis para la etapa 113 también se puede señalar en el flujo de bits a través de la codificación de entropía como se analizó anteriormente.

60 En la etapa 115, el filtrado se realiza en los fotogramas de la señal de vídeo reconstruida de una manera similar a la etapa 107 en el codificador. Por ejemplo, los filtros de supresión de ruido, filtros de desbloqueo, filtros de bucle adaptativo y filtros SAO pueden aplicarse a los fotogramas para eliminar artefactos de bloqueo. Una vez que se filtran los fotogramas, la señal de vídeo se puede enviar a una pantalla en la etapa 117 para que la vea un usuario final.

65 La FIG. 2 es un diagrama esquemático de un sistema 200 de codificación y decodificación (códec) de ejemplo para codificación de vídeo. Específicamente, el sistema 200 de códec proporciona funcionalidad para soportar

la implementación del método 100 de operación. El sistema 200 de códec se generaliza para representar componentes empleados tanto en un codificador como en un decodificador. El sistema 200 de códec recibe y particiona una señal de vídeo como se analiza con respecto a las etapas 101 y 103 en el método 100 de operación, lo que da como resultado una señal 201 de vídeo particionada. Luego, El sistema 200 de códec comprime la señal 201 de vídeo particionada en un flujo de bits codificado cuando actúa como un codificador como se analizó con respecto a las etapas 105, 107 y 109 en el método 100. Cuando actúa como de decodificador, un sistema 200 de códec genera una señal de vídeo de salida del flujo de bits como se analiza con respecto a las etapas 111, 113, 115 y 117 en el método 100 de operación. El sistema 200 de códec incluye un componente 211 de control de codificador general, un componente 213 de escalado y cuantificación de transformación, un componente 215 de estimación intraimagen, un componente 217 de predicción intraimagen, un componente 219 de compensación de movimiento, un componente 221 de estimación de movimiento, un componente 229 de escalado y transformación inversa, un componente 227 de análisis de control de filtros, un componente 225 de filtro en bucle, un componente 223 de búfer de imágenes decodificadas, y un componente 231 de formateo de cabecera y de codificación aritmética binaria adaptativa al contexto (CABAC). Tales componentes se acoplan como se muestra. En la FIG. 2, las líneas negras indican el movimiento de los datos a codificar/decodificar, mientras que las líneas discontinuas indican el movimiento de los datos de control que controlan el funcionamiento de otros componentes. Todos los componentes del sistema 200 de códec pueden estar presentes en el codificador. El decodificador puede incluir un subconjunto de los componentes del sistema 200 de códec. Por ejemplo, el decodificador puede incluir el componente 217 de predicción intraimagen, el componente 219 de compensación de movimiento, el componente de escalamiento y transformación inversa 229, el componente 225 de filtros en bucle y el componente 223 de búfer de imágenes decodificadas. Estos componentes se describen ahora.

La señal 201 de vídeo particionada es una secuencia de vídeo capturada que ha sido particionada en bloques de píxeles por un árbol de codificación. Un árbol de codificación emplea varios modos de división para subdividir un bloque de píxeles en bloques más pequeños de píxeles. Luego, estos bloques pueden subdividirse en bloques más pequeños. Los bloques pueden denominarse nodos en el árbol de codificación. Los nodos padres más grandes se dividen en nodos hijos más pequeños. El número de veces que se subdivide un nodo se denomina profundidad del árbol de codificación/nodo. Los bloques divididos se incluyen unidades de codificación (CU) en algunos casos. Por ejemplo, una CU puede ser una subparte de una CTU que contiene un bloque de luma, uno o más bloque(s) de croma de diferencia roja (Cr) y uno o más bloque(s) de croma de diferencia azul (Cb) junto con las instrucciones de sintaxis correspondientes para la CU. Los modos de división pueden incluir un árbol binario (BT), un árbol triple (TT) y un árbol cuádruple (QT) empleados para particionar un nodo en dos, tres o cuatro nodos hijos, respectivamente, de formas variables dependiendo de los modos de división empleados. La señal 201 de vídeo particionada se reenvía al componente 211 de control de codificador general, al componente 213 de escalado y cuantificación de transformación, al componente 215 de estimación intraimagen, al componente 227 de análisis de control de filtros y al componente 221 de estimación de movimiento para su compresión.

El componente 211 de control de codificador general está configurado para tomar decisiones relacionadas con la codificación de las imágenes de la secuencia de vídeo en el flujo de bits según las restricciones de la aplicación. Por ejemplo, el componente 211 de control de codificador general gestiona la optimización de la tasa de bits/tamaño de flujo de bits frente a la calidad de reconstrucción. Tales decisiones pueden tomarse basadas en la disponibilidad de espacio de almacenamiento/ancho de banda y las solicitudes de resolución de imágenes. El componente 211 de control de codificador general también gestiona la utilización del búfer a la luz de la velocidad de transmisión para mitigar los problemas de infrautilización y desbordamiento del búfer. Para gestionar estos problemas, el componente 211 de control de codificador general gestiona el particionamiento, la predicción y el filtrado por los otros componentes. Por ejemplo, el componente 211 de control de codificador general puede aumentar dinámicamente la complejidad de compresión para aumentar la resolución y aumentar el uso del ancho de banda o disminuir la complejidad de compresión para disminuir la resolución y el uso del ancho de banda. Por lo tanto, el componente 211 de control de codificador general controla los otros componentes del sistema 200 de códec para equilibrar la calidad de reconstrucción de la señal de vídeo con las cuestiones de la velocidad de bits. El componente 211 de control de codificador general crea datos de control, que controlan el funcionamiento de los otros componentes. Los datos de control también se reenvían al componente 231 de formateo de cabecera y CABAC para codificarse en el flujo de bits para señalar parámetros de decodificación en el decodificador.

La señal 201 de vídeo particionada también se envía al componente 221 de estimación de movimiento y al componente 219 de compensación de movimiento para la interpredicción. El fotograma o segmento de la señal 201 de vídeo particionada se puede dividir en múltiples bloques de vídeo. El componente 221 de estimación de movimiento y la unidad 219 de compensación de movimiento realizan una codificación interpredictiva del bloque de vídeo recibido en relación con uno o más bloques en uno o más fotogramas de referencia para proporcionar una predicción temporal. El codificador 200 de vídeo puede realizar múltiples pasadas de codificación, por ejemplo, para seleccionar un modo de codificación adecuado para cada bloque de datos de vídeo.

El componente 221 de estimación de movimiento y el componente 219 de compensación de movimiento pueden estar altamente integrados, pero se ilustran por separado con fines conceptuales. La estimación de movimiento, realizada por el componente 221 de estimación de movimiento, es el proceso de generar vectores de movimiento, que estiman el movimiento para bloques de vídeo. Un vector de movimiento, por ejemplo, puede indicar el desplazamiento de un objeto codificado en relación con un bloque predictivo. Un bloque predictivo es un bloque que, en términos de diferencia de píxeles, coincide estrechamente con el bloque a codificar. Un bloque predictivo también puede denominarse bloque de referencia. Tal diferencia de píxeles puede determinarse por la suma de la diferencia absoluta (SAD), la suma de la diferencia cuadrática (SSD) u otras métricas de diferencia. La HEVC emplea varios objetos codificados, incluidos una CTU, bloques de árbol de codificación (CTB) y CU. Por ejemplo, una CTU se puede dividir en CTB, que luego se pueden dividir en CB para su inclusión en las CU. Una CU se puede codificar como una unidad de predicción (PU) que contiene datos de predicción y/o una unidad de transformación (TU) que contiene datos residuales transformados para la CU. El componente 221 de estimación de movimiento genera vectores de movimiento, las PU y TU usando un análisis de distorsión de tasa como parte de un proceso de optimización de distorsión de tasa. Por ejemplo, el componente 221 de estimación de movimiento puede determinar múltiples bloques de referencia, múltiples vectores de movimiento, etc. para un bloque/fotograma actual, y puede seleccionar los bloques de referencia, vectores de movimiento, etc. que tengan las mejores características de tasa-distorsión. Las mejores características de tasa-distorsión equilibran tanto la calidad de la reconstrucción del vídeo (por ejemplo, cantidad de pérdida de datos por compresión) con la eficiencia de la codificación (por ejemplo, el tamaño de la codificación final).

En algunos ejemplos, el sistema 200 de códec puede calcular valores para posiciones de píxeles de subenteros de imágenes de referencia almacenadas en el componente 223 de búfer de imágenes decodificadas. Por ejemplo, el codificador 200 de vídeo puede interpolar valores de posiciones de un cuarto de píxel, posiciones de un octavo de píxel u otras posiciones fraccionarias de píxel de la imagen de referencia. Por lo tanto, el componente 221 de estimación de movimiento puede realizar una búsqueda de movimiento con respecto a las posiciones de píxeles completos y posiciones de píxeles fraccionarios y generar un vector de movimiento con precisión de píxeles fraccionarios. El componente 221 de estimación de movimiento calcula un vector de movimiento para una PU de un bloque de vídeo en un segmento intercodificado comparando la posición de la PU con la posición de un bloque predictivo de una imagen de referencia. El componente 221 de estimación de movimiento genera el vector de movimiento calculado como datos de movimiento para el componente 231 de formato de cabecera y CABAC para la codificación y el movimiento para el componente 219 de compensación de movimiento.

La compensación de movimiento, realizada por el componente 219 de compensación de movimiento, puede implicar buscar o generar el bloque predictivo basándose en el vector de movimiento determinado por el componente 221 de estimación de movimiento. Nuevamente, el componente 221 de estimación de movimiento y el componente 219 de compensación de movimiento pueden integrarse funcionalmente, en algunos ejemplos. Al recibir el vector de movimiento para la PU del bloque de vídeo actual, el componente 219 de compensación de movimiento puede ubicar el bloque predictivo al que apunta el vector de movimiento. Luego, se forma un bloque de vídeo residual restando los valores de píxel del bloque predictivo de los valores de píxel del bloque de vídeo actual que se está codificando, formando valores de diferencia de píxel. En general, el componente 221 de estimación de movimiento realiza la estimación de movimiento en relación con los componentes de luma, y la unidad 219 de compensación de movimiento usa vectores de movimiento calculados basándose en los componentes de luma tanto para los componentes de croma como para los componentes de luma. El bloque predictivo y el bloque residual se reenvían para transformar el componente 213 de escalado y cuantificación.

La señal 201 de vídeo particionada también se envía al componente 215 de estimación intraimagen y al componente 217 de predicción intraimagen. Al igual que con el componente 221 de estimación de movimiento y el componente 219 de compensación de movimiento, el componente 215 de estimación intraimagen y el componente 217 de predicción intraimagen pueden estar altamente integrados, pero se ilustran por separado con fines conceptuales. El componente 215 de estimación intraimagen y el componente 217 de predicción intraimagen intrapredicen un bloque actual en relación con bloques en un fotograma actual, como una alternativa a la interpredicción realizada por el componente 221 de estimación de movimiento y el componente 219 de compensación de movimiento entre fotogramas, como se describió anteriormente. En particular, el componente 215 de estimación de intraimagen determina un modo de intrapredicción a usar para codificar un bloque actual. En algunos ejemplos, el componente 215 de estimación intraimagen selecciona un modo de intrapredicción adecuado para codificar un bloque actual a partir de múltiples modos de intrapredicción probados. Luego, los modos de intrapredicción seleccionados se reenvían al componente 231 de formato de cabecera y CABAC para su codificación.

Por ejemplo, el componente 215 de estimación intraimagen calcula los valores de tasa-distorsión usando un análisis de tasa-distorsión para los varios modos de intrapredicción probados, y selecciona el modo de intrapredicción que tenga las mejores características tasa-distorsión entre los modos probados. El análisis de tasa-distorsión generalmente determina una cantidad de distorsión (o error) entre un bloque codificado y un

bloque original sin codificar que fue codificado para producir el bloque codificado, así como una tasa de bits (es decir, un número de bits) usada para producir el bloque codificado. El componente 215 de estimación intraimagen calcula relaciones a partir de las relaciones de distorsiones y tasas para los varios bloques codificados para determinar qué modo de intrapredicción presenta el mejor tasa-distorsión para el bloque. Además, el componente 215 de estimación intraimagen puede configurarse para codificar bloques de profundidad de un mapa de profundidad usando un modo de modelado de profundidad (DMM) basado en la optimización tasa-distorsión (RDO).

El componente 217 de predicción intraimagen puede generar un bloque residual a partir del bloque predictivo basándose en los modos de intrapredicción seleccionados determinados por el componente 215 de estimación intraimagen cuando se implementa en un codificador o leer el bloque residual del flujo de bits cuando se implementa en un decodificador. El bloque residual incluye la diferencia en valores entre el bloque predictivo y el bloque original, representado como una matriz. El bloque predictivo y el bloque residual se reenvían al componente 213 de escalado y cuantificación de transformación. El componente 215 de estimación intraimagen y el componente 217 de predicción intraimagen pueden operar tanto en componentes de luma como de croma.

El componente 213 de escalado y cuantificación de transformación está configurado para comprimir adicionalmente el bloque residual. El componente 213 de escalado y cuantificación de transformación aplica una transformación, tal como una transformación de coseno discreta (DCT), una transformación de seno discreta (DST) o una transformación conceptualmente similar, al bloque residual, produciendo un bloque de vídeo que comprende valores de coeficiente de transformación residual. También podrían usarse transformadas de ondícula, transformadas de enteros, transformadas de subbanda u otros tipos de transformaciones. La transformación puede convertir la información residual de un dominio de valor de píxel a un dominio de transformación, tal como un dominio de frecuencia. El componente 213 de escalado y cuantificación de transformación también está configurado para escalar la información residual de transformación, por ejemplo basándose en la frecuencia. Tal escalado implica aplicar un factor de escala a la información residual para que la información de diferente frecuencia se cuantifique en diferentes granularidades, lo que puede afectar la calidad visual final del vídeo reconstruido. El componente 213 de cuantificación y escalado de transformación también está configurado para cuantificar los coeficientes de transformación para reducir adicionalmente la tasa de bits. El proceso de cuantificación puede reducir la profundidad de bit asociada con algunos o todos los coeficientes. El grado de cuantificación puede modificarse ajustando un parámetro de cuantificación. En algunos ejemplos, luego, el componente 213 de cuantificación y escalado de transformación puede realizar un escaneo de la matriz que incluya los coeficientes de transformación cuantificados. Los coeficientes de transformación cuantificados se envían al componente 231 de formateo de cabecera y CABAC para codificarse en el flujo de bits.

El componente 229 de escalado y transformación inversa aplica una operación inversa del componente 213 de escalado y cuantificación de transformación para soportar la estimación de movimiento. El componente 229 de escalado y transformación inversa aplica escalado inverso, transformación y/o cuantización para reconstruir el bloque residual en el dominio de píxeles, por ejemplo, para su uso posterior como un bloque de referencia que puede convertirse en un bloque predictivo para otro bloque actual. El componente 221 de estimación de movimiento y/o el componente 219 de compensación de movimiento pueden calcular un bloque de referencia añadiendo el bloque residual de nuevo a un bloque predictivo correspondiente para su uso en la estimación de movimiento de un bloque/fotograma posterior. Los filtros se aplican a los bloques de referencia reconstruidos para mitigar los artefactos creados durante el escalado, cuantificación y transformación. De lo contrario, tales artefactos podrían causar una predicción inexacta (y crear artefactos adicionales) cuando se predicen bloques posteriores.

El componente 227 de análisis de control de filtros y el componente 225 de filtros en bucle aplican los filtros a los bloques residuales y/o a los bloques de imagen reconstruidos. Por ejemplo, el bloque residual transformado del componente 229 de escalado y transformación inversa puede combinarse con un bloque de predicción correspondiente del componente 217 de predicción intraimagen y/o el componente 219 de compensación de movimiento para reconstruir el bloque de imagen original. Luego, los filtros pueden aplicarse al bloque de imagen reconstruido. En algunos ejemplos, los filtros pueden aplicarse en cambio a los bloques residuales. Al igual que con otros componentes de la FIG. 2, el componente 227 de análisis de control de filtros y el componente 225 de filtros en bucle están altamente integrados y pueden implementarse juntos, pero se representan por separado con fines conceptuales. Los filtros aplicados a los bloques de referencia reconstruidos se aplican a regiones espaciales particulares e incluyen múltiples parámetros para ajustar cómo se aplican tales filtros. El componente 227 de análisis de control de filtro analiza los bloques de referencia reconstruidos para determinar dónde se deben aplicar tales filtros y establece los parámetros correspondientes. Estos datos se reenvían al componente 231 de formato de cabecera y CABAC como datos de control de filtro para codificación. El componente 225 de filtros en bucle aplica tales filtros basándose en los datos de control del filtro. Los filtros pueden incluir un filtro de desbloqueo, un filtro de supresión de ruido, un filtro SAO y un filtro de bucle adaptativo. Tales filtros pueden aplicarse en el dominio espacial/de píxeles (por ejemplo, en un bloque de píxeles reconstruido) o en el dominio de la frecuencia, dependiendo del ejemplo.

5 Cuando funciona como un codificador, el bloque de imagen reconstruido filtrado, el bloque residual y/o el bloque de predicción se almacenan en el componente 223 de búfer de imágenes decodificadas para su uso posterior en la estimación de movimiento como se analizó anteriormente. Cuando funciona como un decodificador, el componente 223 de búfer de imágenes decodificadas almacena y reenvía los bloques reconstruidos y filtrados hacia una pantalla como parte de una señal de vídeo de salida. El componente 223 de búfer de imágenes decodificadas puede ser cualquier dispositivo de memoria capaz de almacenar bloques de predicción, bloques residuales y/o bloques de imagen reconstruidos.

10 El componente 231 de formateo de cabecera y CABAC recibe los datos de los varios componentes del sistema 200 de códec y codifica tales datos en un flujo de bits codificado para su transmisión hacia un decodificador. Específicamente, el componente 231 de formateo de cabecera y CABAC genera varias cabeceras para codificar datos de control, tal como datos de control general y datos de control de filtro. Además, los datos de predicción, incluidos los datos de intrapredicción y de movimiento, así como los datos residuales en la forma de datos de coeficientes de transformación cuantificados, se codifican todos en el flujo de bits. El flujo de bits final incluye toda la información deseada por el decodificador para reconstruir la señal 201 de vídeo particionada original. Tal información también puede incluir tablas de índice de modo de intrapredicción (también denominadas tablas de mapeo de palabras de código), definiciones de contextos de codificación para varios bloques, indicaciones de los modos de intrapredicción más probables, una indicación de información de partición, etc. Tales datos pueden codificarse empleando codificación de entropía. Por ejemplo, la información puede codificarse empleando codificación de longitud variable adaptable al contexto (CAVLC), CABAC, codificación aritmética binaria adaptable al contexto basada en sintaxis (SBAC), codificación de entropía de particionamiento de intervalo de probabilidad (PIPE) u otra técnica de codificación de entropía. Después de la codificación de entropía, el flujo de bits codificado puede transmitirse a otro dispositivo (por ejemplo, un decodificador de vídeo) o archivar para su posterior transmisión o recuperación.

30 La FIG. 3 es un diagrama de bloques que ilustra un codificador 300 de vídeo de ejemplo. El codificador 300 de vídeo se puede emplear para implementar las funciones de codificación del sistema 200 de códec y/o implementar las etapas 101, 103, 105, 107 y/o 109 del método 100. El codificador 300 de vídeo particiona una señal de vídeo de entrada, lo da como resultado una señal 301 de vídeo particionada, que es sustancialmente similar a la señal 201 de vídeo particionada. Luego, la señal 301 de vídeo particionada es comprimida y codificada en un flujo de bits por componentes del codificador 300.

35 Específicamente, la señal 301 de vídeo particionada se reenvía a un componente 317 de predicción intraimagen para la intrapredicción. El componente 317 de predicción intraimagen puede ser sustancialmente similar al componente 215 de estimación intraimagen y al componente 217 de predicción intraimagen. La señal 301 de vídeo particionada también se reenvía a un componente 321 de compensación de movimiento para la interpredicción basándose en bloques de referencia en un búfer 323 de imágenes decodificadas. El componente 321 de compensación de movimiento puede ser sustancialmente similar al componente 221 de estimación de movimiento y al componente 219 de compensación de movimiento. Los bloques de predicción y los bloques residuales del componente 317 de predicción intraimagen y del componente 321 de compensación de movimiento se reenvían a un componente 313 de transformación y cuantificación para la transformación y cuantificación de los bloques residuales. El componente 313 de cuantificación y transformación inversa puede ser sustancialmente similar al componente 213 de escalado y cuantificación de transformación. Los bloques residuales transformados y cuantificados y los bloques de predicción correspondientes (junto con los datos de control asociados) se reenvían a un componente 331 de codificación de entropía para codificación en un flujo de bits. El componente 331 de codificación de entropía puede ser sustancialmente similar al componente 231 de formateo de cabecera y CABAC.

50 Los bloques residuales transformados y cuantificados y/o los bloques de predicción correspondientes también se reenvían desde el componente 313 de transformación y cuantificación a un componente 329 de cuantificación y transformación inversa para la reconstrucción en bloques de referencia para su uso por el componente 321 de compensación de movimiento. El componente 329 de cuantificación y transformación inversa puede ser sustancialmente similar al componente 229 de escalado y transformación inversa. Los filtros en bucle en un componente 325 de filtros en bucle también se aplican a los bloques residuales y/o bloques de referencia reconstruidos, dependiendo del ejemplo. El componente 325 de filtros en bucle puede ser sustancialmente similar al componente 227 de análisis de control de filtros y al componente 225 de filtros en bucle. El componente 325 de filtros en bucle puede incluir múltiples filtros como se analizó con respecto al componente 225 de filtros en bucle. Los bloques filtrados se almacenan entonces en un componente 323 de búfer de imágenes decodificadas para su uso como bloques de referencia por el componente 321 de compensación de movimiento. El componente 323 de búfer de imágenes decodificadas puede ser sustancialmente similar al componente 223 de búfer de imágenes decodificadas.

65 La FIG. 4 es un diagrama de bloques que ilustra un decodificador 400 de vídeo de ejemplo. El decodificador 400 de vídeo puede ser empleado para implementar funciones de decodificación del sistema 200 de códec y/o implementar las etapas 111, 113, 115 y/o 117 del método 100 de operación. El decodificador 400 recibe un

flujo de bits, por ejemplo de un codificador 300, y genera una señal de vídeo de salida reconstruida basándose en el flujo de bits para visualizarla a un usuario final.

5 El flujo de bits es recibido por un componente 433 de decodificación de entropía. El componente 433 de decodificación de entropía está configurado para implementar un esquema de decodificación de entropía, tal como CAVLC, CABAC, SBAC, codificación PIPE u otras técnicas de codificación de entropía. Por ejemplo, el componente 433 de decodificación de entropía puede emplear información de cabecera para proporcionar un contexto para interpretar datos adicionales codificados como palabras de código en el flujo de bits. La información decodificada incluye cualquier información deseada para decodificar la señal de vídeo, tal como  
10 datos de control general, datos de control de filtro, información de partición, datos de movimiento, datos de predicción y coeficientes de transformación cuantificados de bloques residuales. Los coeficientes de transformación cuantificados se reenvían a un componente 429 de cuantificación y transformación inversa para la reconstrucción en bloques residuales. El componente 429 de cuantificación y transformación inversa puede ser sustancialmente similar al componente 329 de cuantificación y transformación inversa.

15 Los bloques residuales reconstruidos y/o los bloques de predicción se reenvían al componente 417 de predicción intraimagen para la reconstrucción en bloques de imagen basándose en operaciones de intrapredicción. El componente 417 de predicción intraimagen puede ser sustancialmente similar al componente 215 de estimación intraimagen y al componente 217 de predicción intraimagen. Específicamente,  
20 el componente 417 de predicción intraimagen emplea modos de predicción para localizar un bloque de referencia en un fotograma y aplica un bloque residual al resultado para reconstruir bloques de imagen intrapredicha. Los bloques de imagen intrapredicha reconstruidos y/o los bloques residuales y los datos de interpredicción correspondientes se reenvían a un componente 423 de búfer de imágenes decodificadas a través del componente 425 de filtros en bucle, que puede ser sustancialmente similar al componente 223 de  
25 búfer de imágenes decodificadas y al componente 225 de filtros en bucle, respectivamente. El componente 425 de filtros en bucle filtra los bloques de imagen reconstruidos, bloques residuales y/o bloques de predicción, y tal información se almacena en el componente 423 de búfer de imágenes decodificadas. Los bloques de imagen reconstruidos del componente 423 de búfer de imágenes decodificadas se reenvían a un componente 421 de compensación de movimiento para la interpredicción. El componente 421 de compensación de movimiento puede ser sustancialmente similar al componente 221 de estimación de movimiento y al  
30 componente 219 de compensación de movimiento. Específicamente, el componente 421 de compensación de movimiento emplea vectores de movimiento de un bloque de referencia para generar un bloque de predicción y aplica un bloque residual al resultado para reconstruir un bloque de imagen. Los bloques reconstruidos resultantes también pueden reenviarse a través del componente 425 de filtros en bucle al componente 423 de  
35 búfer de imágenes decodificadas. El componente 423 de búfer de imágenes decodificadas continúa almacenando bloques de imagen reconstruidos adicionales, que pueden reconstruirse en fotogramas a través de la información de partición. Tales fotogramas también pueden colocarse en una secuencia. La secuencia es enviada hacia una pantalla como una señal de vídeo de salida reconstruida.

40 La FIG. 5 es un diagrama esquemático que ilustra un flujo de bits 500 de ejemplo que contiene una secuencia de vídeo codificada. Por ejemplo, el flujo de bits 500 puede ser generado por un sistema 200 de códec y/o un codificador 300 para su decodificación por un sistema 200 de códec y/o un decodificador 400. Como otro ejemplo, el flujo de bits 500 puede ser generado por un codificador en la etapa 109 del método 100 para su uso por un decodificador en la etapa 111.

45 El flujo de bits 500 incluye un conjunto de parámetros de secuencia (SPS) 510, una pluralidad de conjuntos de parámetros de imagen (PPS) 512, una pluralidad de cabeceras 514 de segmento y datos 520 de imagen. Un SPS 510 contiene datos de secuencia comunes a todas las imágenes de la secuencia de vídeo contenida en el flujo de bits 500. Tales datos pueden incluir el tamaño de la imagen, la profundidad de bits, los parámetros de la herramienta de codificación, las restricciones de tasa de bits, etc. El PPS 512 contiene parámetros que son específicos de cada imagen. Por lo tanto, puede haber un PPS 512 por imagen en la secuencia de vídeo. El PPS 512 puede indicar herramientas de codificación disponibles para segmentos en imágenes correspondientes, parámetros de cuantificación, desplazamientos, parámetros de herramientas de codificación  
50 específicos de la imagen (por ejemplo, controles de filtro), etc. La cabecera 514 del segmento contiene parámetros que son específicos para cada segmento en una imagen. Por lo tanto, puede haber una cabecera 514 de segmento por segmento en la secuencia de vídeo. La cabecera 514 de segmento puede contener información de tipo de segmento, recuentos de orden de imagen (POC), listas de imágenes de referencia, ponderaciones de predicción, puntos de entrada de mosaico, parámetros de desbloqueo, etc.

60 Los datos 520 de imagen contienen datos de vídeo codificados según interpredicción y/o intrapredicción, así como datos residuales transformados y cuantificados correspondientes. Tales datos 520 de imagen se ordenan según el particionamiento usado para particionar la imagen antes de la codificación. Por ejemplo, la imagen en los datos 520 de imagen se divide en segmentos 521. Cada segmento 521 se divide además en mosaicos 523. Los mosaicos 523 se dividen además en CTU 527. Las CTU 527 se dividen además en bloques de codificación basándose en árboles de codificación. Luego, los bloques de codificación pueden codificarse/decodificarse según mecanismos de predicción. Una imagen/fotografía puede contener uno o más fragmentos 521. Se  
65

emplea una cabecera 514 de segmento por segmento 521. Cada segmento 521 puede contener uno o más mosaicos 523, que a su vez pueden contener una pluralidad de CTU 527.

Cada segmento 521 puede ser un rectángulo definido por un mosaico 523 en una esquina superior izquierda y un mosaico 523 en una esquina inferior derecha. A diferencia de otros sistemas de codificación, un segmento 521 puede no atravesar todo el ancho de una imagen. Un segmento 521 es la unidad más pequeña que un decodificador puede visualizar por separado. Por lo tanto, dividir los segmentos 521 en unidades más pequeñas permite generar subimágenes de una manera que es lo suficientemente granular para visualizar las áreas deseadas de una imagen. Por ejemplo, en un contexto de realidad virtual (VR), una imagen puede contener toda una esfera visible de datos, pero un usuario solo puede ver una subimagen en una pantalla montada en la cabeza. Los segmentos 521 más pequeños permiten que tales subimágenes se señalicen por separado. Un segmento 521 también se señala generalmente en una unidad 533 NAL VCL separada. Además, un segmento 521 puede no permitir la predicción basándose en otros segmentos 521, lo que permite que cada segmento 521 se codifique independientemente de otros segmentos 521.

Los segmentos 521 se particionan en un número entero de mosaicos 523. Un mosaico 523 es una parte particionada de un segmento 521 creada por límites horizontales y verticales. Los mosaicos 523 pueden codificarse en orden de escaneo de trama y pueden permitir o no la predicción basándose en otros mosaicos 523, dependiendo del ejemplo. Cada mosaico 523 puede tener un ID 524 de mosaico único en la imagen. Un ID 524 de mosaico es un identificador numérico que se puede usar para distinguir un mosaico 523 de otro. Un ID 524 de mosaico puede tomar el valor de un índice de mosaico que aumenta numéricamente en el orden de escaneo de trama. El orden de escaneo de trama es de izquierda a derecha y de arriba a abajo. Los ID 524 de mosaico también pueden emplear otros valores numéricos. Sin embargo, el ID 524 de mosaico siempre debe aumentar en el orden de escaneo de trama para soportar los cálculos analizados en la presente memoria. Por ejemplo, los límites del segmento 521 se pueden determinar según el ID 524 de mosaico del segmento 523 en la esquina superior izquierda del segmento 521 y el ID 524 de mosaico del segmento 523 en la esquina inferior derecha del segmento 521. Cuando el ID 524 de mosaico es un valor diferente de un índice de mosaico, se puede señalar un mecanismo de conversión en el flujo de bits 500, por ejemplo en el PPS 512. Además, cada mosaico 523 puede estar asociado con un desplazamiento 525 de punto de entrada. El desplazamiento 525 de punto de entrada indica la ubicación del primer bit de datos codificados asociado con el mosaico 523. El primer mosaico 523 puede tener un desplazamiento 525 de punto de entrada de cero y los mosaicos 523 adicionales pueden tener cada uno un desplazamiento 525 de punto de entrada igual al número de bits de datos codificados en los mosaicos 523 anteriores. Como tal, se puede inferir que el número de desplazamientos 525 de puntos de entrada es uno menos que el número de mosaicos 523.

Los mosaicos 523 se dividen además en las CTU 527. Una CTU 527 es una subparte de un mosaico 523 que puede subdividirse adicionalmente por una estructura de árbol de codificación en bloques de codificación que pueden codificarse por un codificador y decodificarse por un decodificador. Las CTU 527 están asociadas cada una a una dirección 529 de CTU. Una dirección 529 de CTU denota la ubicación de una CTU 527 correspondiente en el flujo de bits 500. Específicamente, una dirección 529 de CTU puede denotar la ubicación de una CTU 527 correspondiente en una unidad 533 NAL VCL. En algunos ejemplos, las direcciones 529 de CTU para las CTU 527 pueden estar señalizadas explícitamente, por ejemplo en el PPS 512. En otros ejemplos, el decodificador puede derivar las direcciones 529 de CTU. Por ejemplo, las direcciones 529 de CTU se pueden asignar basándose en el ID 524 de mosaico del mosaico 523 que contiene las CTU 527 correspondientes. En tal caso, el decodificador puede determinar los mosaicos 523 en un segmento 521 basándose en los ID 524 de mosaico de los mosaicos 523 superior izquierdo e inferior derecho. Luego, el decodificador puede usar los mosaicos 523 determinados en el segmento 521 para determinar el número de CTU 527 en el segmento 521. Además, el decodificador puede usar los ID 524 de mosaico conocidos y el número de CTU 527 para determinar las direcciones 529 de CTU. Además, como el decodificador conoce el número de CTU 527, se puede omitir un indicador que indica si cada CTU 527 es la última CTU 527 en una unidad 533 NAL VCL. Esto se debe a que el decodificador puede determinar qué CTU 527 es la última CTU 527 en una unidad 533 NAL VCL al conocer el número de CTU 527 en el segmento 521, que está contenida en la unidad 533 NAL VCL. Sin embargo, se puede colocar un bit de relleno después de la última CTU 527 en un mosaico 523 para ayudar a distinguir entre los mosaicos 523 en algunos ejemplos. Como se puede observar, la señalización de los límites del segmento 521 basándose en los ID 524 de mosaico puede permitir que el decodificador infiera una cantidad significativa de datos, que luego se pueden omitir del flujo de bits 500 para aumentar la eficiencia de la codificación.

El flujo de bits 500 se posiciona en unidades 533 NAL VCL y unidades 531 NAL no VCL. Una unidad NAL es una unidad de datos codificados dimensionada para ser colocada como carga útil para un único paquete para su transmisión a través de una red. Una unidad 533 NAL VCL es una unidad NAL que contiene datos de vídeo codificados. Por ejemplo, cada unidad 533 NAL VCL puede contener un segmento 521 de datos que incluye los correspondientes mosaicos 523, las CTU 527 y bloques de codificación. Una unidad 531 NAL no VCL es una unidad NAL que contiene sintaxis de soporte, pero no contiene datos de vídeo codificados. Por ejemplo, una unidad 531 NAL no VCL puede contener el SPS 510, un PPS 512, una cabecera 514 de segmento, etc. Como tal, el decodificador recibe el flujo de bits 500 en la unidad 533 NAL VCL discreta y las unidades 531

- NAL no VCL. En aplicaciones de transmisión en vivo, el decodificador puede decodificar datos de vídeo actuales sin esperar a que se reciba todo el flujo de bits 500. Como tal, los ID 524 de mosaico, los desplazamientos 525 de puntos de entrada y las direcciones 529 de CTU permiten al decodificador ubicar correctamente los datos de vídeo en la unidad 533 NAL VCL para una decodificación rápida, procesamiento paralelo y otros mecanismos de visualización de vídeo. En consecuencia, el cálculo de los ID 524 de mosaico, los desplazamientos 525 de puntos de entrada y/o las direcciones 529 de CTU permite la implementación de mecanismos de decodificación y visualización eficientes al tiempo que reduce el tamaño del flujo de bits 500 y, por lo tanto, aumenta la eficiencia de codificación.
- La FIG. 6 es un diagrama esquemático que ilustra una imagen 600 de ejemplo particionada para codificación. Por ejemplo, una imagen 600 puede codificarse y decodificarse a partir de un flujo de bits 500, por ejemplo, por un sistema 200 de códec, un codificador 300 y/o un decodificador 400. Además, la imagen 600 se puede particionar para soportar la codificación y decodificación según el método 100.
- La imagen 600 se puede particionar en segmentos 621, mosaicos 623 y las CTU 627, que pueden ser sustancialmente similares a los segmentos 521, mosaicos 523 y las CTU 527, respectivamente. En la FIG. 6, los segmentos 621 están representados por líneas en negrita con fondos blancos alternativos y rayas para diferenciar gráficamente entre los segmentos 621. Los mosaicos 623 se muestran con líneas discontinuas. Los límites del mosaico 623 ubicados en los límites del segmento 621 se representan como líneas discontinuas en negrita y los límites del mosaico 623 que no están ubicados en los límites del segmento 621 se representan como líneas discontinuas sin negrita. Los límites de CTU 627 se representan como líneas sólidas, sin negrita, excepto en las ubicaciones donde los límites de CTU 627 están cubiertos por los límites del mosaico 623 o del segmento 621. En este ejemplo, la imagen 600 incluye nueve segmentos 621, veinticuatro mosaicos 623 y doscientos dieciséis CTU 627.
- Como se muestra, un segmento 621 es un rectángulo con límites que pueden definirse por los mosaicos 623 incluidos. Es posible que el segmento 621 no se extienda por todo el ancho de la imagen 600. Los mosaicos 623 se pueden generar en los segmentos 621 según filas y columnas. Luego, las CTU 627 pueden ser particionadas de los mosaicos 623 para crear particiones de imagen 600 adecuadas para ser subdivididas en bloques de codificación para codificar según la interpredicción y/o la intrapredicción.
- Al emplear lo anterior, se pueden mejorar los sistemas de codificación de vídeo. Por ejemplo, los segmentos están diseñados de manera que las CTU contenidas en un segmento pueden no ser simplemente un conjunto de CTU de una imagen siguiendo un orden de escaneo de trama de CTU de la imagen. En cambio, un segmento se define como un conjunto de CTU que cubren una región rectangular de una imagen. Además, cada segmento está en su propia unidad NAL. Además, las direcciones de las CTU contenidas en un segmento se pueden señalar señalizando, en la cabecera de segmento, las direcciones de las CTU en orden de escaneo de trama de las CTU superiores izquierda e inferiores derechas en el segmento. Además, los segmentos están diseñados para contener, y solo contener, un conjunto de mosaicos completos que cubren una región rectangular de una imagen. Cada segmento está en su propia unidad NAL. De esta manera, el propósito de tener más de un segmento puede ser colocar un conjunto de mosaicos que cubren una región rectangular de una imagen en una unidad NAL. En algunos casos, hay uno o más segmentos en una imagen, y cada uno de estos segmentos puede contener un conjunto de mosaicos completos que cubren una región rectangular. También puede haber otro segmento en la imagen que cubra el resto de los mosaicos de la imagen. La región cubierta por este segmento puede ser una región rectangular con un agujero que esté cubierto por otros segmentos. Por ejemplo, para fines de región de interés, una imagen puede contener dos segmentos, en los que un segmento contiene un conjunto de mosaicos completos que cubren la región de interés y el otro segmento contiene los mosaicos restantes de la imagen.
- Las direcciones de las CTU contenidas en un segmento pueden ser señalizadas explícita o implícitamente por los ID de mosaico contenidos en el segmento. Para una señalización eficiente, en algunos ejemplos solo se pueden señalar los ID de mosaico de los mosaicos superior izquierdo e inferior derecho. Para mejorar adicionalmente la eficiencia de la señalización, se puede señalar un indicador que indique si el segmento contiene un solo mosaico y, en caso afirmativo, solo se puede señalar un ID de mosaico. En otros casos, se señalan todos los ID de mosaico contenidos en un segmento. En algunos ejemplos, los valores de ID de mosaico se asignan para que sea iguales que el índice de mosaico dentro de la imagen. La longitud, en bits, para señalar explícitamente los ID de mosaico en una cabecera de segmento para la derivación de las direcciones de las CTU contenidas en un segmento, se puede derivar según el número de mosaicos en la imagen (por ejemplo, celda de logaritmo dos de la cantidad de mosaicos en una imagen). El número de mosaicos en la imagen se puede señalar explícitamente en un conjunto de parámetros o derivarse según la configuración de mosaicos señalada en un conjunto de parámetros. En algunos ejemplos, la longitud, en bits, para señalar explícitamente los ID de mosaico en una cabecera de segmento para la derivación de las direcciones de las CTU contenidas en un segmento se puede señalar en un conjunto de parámetros. En algunos ejemplos, el número de puntos de entrada, que es igual al número de mosaicos en el segmento menos 1, se deriva y no se señala en la cabecera de segmento. En otro ejemplo, se evita la señalización de un indicador para cada CTU que indica si la CTU es el final de un segmento.

En algunos ejemplos, los segmentos y los mosaicos están diseñados de manera que no es necesario reescribir las cabeceras de los segmentos cuando se extrae un conjunto de mosaicos, tal como conjuntos de mosaicos con restricciones de movimiento (MCTS), de un flujo de bits para crear un subflujo de bits conforme. Por ejemplo, el ID de mosaico se puede señalar explícitamente para cada mosaico en el conjunto de parámetros en el que se señala la configuración de mosaico. Los ID de mosaico son cada uno único dentro de una imagen. Es posible que los ID de mosaico no sean continuos dentro de una imagen. Sin embargo, los ID de mosaico se deben organizar en orden creciente (por ejemplo, aumentando de manera monótona) en la dirección del escaneo de trama de mosaicos de una imagen. Con esto, el orden de decodificación de los segmentos de una imagen se puede restringir para que sea en valor creciente del ID de mosaico superior izquierdo. Cuando el ID de mosaico no se señala explícitamente y se infiere que es igual que el índice del mosaico, se puede usar lo siguiente para señalar los valores de ID de mosaico en las cabeceras de segmento. Se puede señalar un indicador que indique si el segmento es el primer segmento de la imagen. Cuando el indicador indica que el segmento es el primer segmento de la imagen, se puede omitir la señalización del ID de mosaico del mosaico superior izquierdo del segmento, ya que se puede inferir que es el mosaico con el índice de mosaico más bajo (por ejemplo, índice de mosaico cero - asumiendo que el índice de mosaico comienza desde cero).

En otro ejemplo, una imagen puede contener cero, uno o más MCTS. Un MCTS puede contener uno o más mosaicos. Cuando un mosaico en un segmento es parte de un MCTS, el MCTS se restringe para que todos los mosaicos en el segmento sean parte del mismo MCTS. El segmento se puede restringir adicionalmente para que la configuración de mosaico de todas las imágenes que contienen mosaicos de un MCTS sea igual con respecto a las posiciones y tamaños de los mosaicos dentro del MCTS. En algunos ejemplos, el segmento está restringido de manera que un MCTS está contenido exclusivamente en un segmento. Esto tiene dos consecuencias. En este caso, cada MCTS está en una unidad NAL separada. Además, cada MCTS tiene forma rectangular.

La señalización de los MCTS puede ser la siguiente. Se puede señalar un indicador en una cabecera de segmento para indicar si el segmento contiene unidades NAL con un MCTS en la unidad de acceso que contenga el segmento correspondiente. Otra información de soporte para MCTS (por ejemplo, información de perfil, nivel y capa del subflujo de bits resultante de la extracción del MCTS) se señala en un mensaje SEI. Alternativamente, tanto la indicación de la bandera como la información de soporte de MCTS pueden señalizarse en un mensaje SEI. Para permitir la señalización del tratamiento de los límites de MCTS como límites de imagen, se señala un elemento de sintaxis que indica si todos los límites de mosaico se tratan igual que los límites de imagen, por ejemplo, en el conjunto de parámetros en donde se señala la configuración de mosaico. Además, se puede señalar en la cabecera de segmento un elemento de sintaxis que indica si todos los límites de un segmento se tratan igual que los límites de imagen, por ejemplo, cuando otros elementos de sintaxis no indiquen que todos los límites de mosaico se tratan igual que los límites de imagen.

Se puede señalar un elemento de sintaxis que indica si las operaciones de filtrado en bucle se pueden aplicar en cada límite de mosaico solo cuando la sintaxis no indique de otra manera que todos los límites de mosaico se tratan igual que los límites de imagen. En este caso, tratar un límite de mosaico como un límite de imagen indica que, entre otros aspectos, no se pueden aplicar operaciones de filtrado en bucle en cada límite de mosaico. En otros ejemplos, un elemento de sintaxis que indica si las operaciones de filtrado en bucle se pueden aplicar en cada límite de mosaico se señala independientemente de las indicaciones de si todos los límites de mosaico se tratan igual que los límites de imagen. En este caso, tratar un límite de mosaico como límite de imagen indica que las operaciones de filtrado en bucle pueden seguir aplicándose en cada límite de mosaico.

En algunos ejemplos, un límite MCTS se trata como un límite de imagen. Además, el elemento de sintaxis en la cabecera de segmento que indica si el límite de segmento se trata igual que el límite de imagen también se puede hacer condicional al indicador que indica si el segmento contiene MCTS. En algunos casos, el valor de un indicador que indica que el límite de MCTS debe tratarse como un límite de imagen se puede inferir cuando el indicador en la cabecera de segmento indica que el segmento contiene MCTS.

Cuando se indica que los límites de un mosaico o segmento deben tratarse como límites de imagen, se aplica lo siguiente. En el proceso de derivación para la predicción del vector de movimiento de luma temporal, las posiciones de los límites derecho e inferior de la imagen usadas en el proceso, indicadas por `pic_height_in_luma_samples - 1` y `pic_width_in_luma_samples - 1`, respectivamente, se reemplazan con las posiciones de los límites derecho e inferior, respectivamente, del mosaico o segmento, en unidades de muestras de luma. En el proceso de interpolación de muestras de luma, las posiciones de los límites izquierdo, derecho, superior e inferior de la imagen usadas en el proceso, indicadas por 0, `pic_height_in_luma_samples - 1`, 0, `pic_width_in_luma_samples - 1`, respectivamente, se reemplazan con las posiciones de los límites izquierdo, derecho, superior e inferior, respectivamente, del mosaico o segmento, en unidades de muestras de luma, respectivamente. En el proceso de interpolación de muestras de croma, las posiciones de los límites izquierdo, derecho, superior e inferior de la imagen usadas en el proceso, indicadas por 0,

$\text{pic\_height\_in\_luma\_samples} / \text{SubWidthC} - 1, 0, \text{pic\_width\_in\_luma\_samples} / \text{SubWidthC} - 1$ , respectivamente, se reemplazan con las posiciones de los límites izquierdo, derecho, superior e inferior, respectivamente, del mosaico o segmento, en unidades de muestras de croma, respectivamente.

- 5 Los mecanismos anteriores se pueden implementar de la siguiente manera. Un segmento se define como un número entero de mosaicos que cubren una región rectangular de una imagen y que están contenidos exclusivamente en una única unidad NAL. Una cabecera de segmento se define como una parte de un segmento codificado que contiene los elementos de datos pertenecientes a todos los mosaicos representados en el segmento. Un mosaico se define como una región rectangular de CTU dentro de una columna de
- 10 mosaicos particular y una fila de mosaicos particular en una imagen. Una columna de mosaicos se define como una región rectangular de CTU que tiene una altura igual a la altura de la imagen y un ancho especificado por elementos de sintaxis en el conjunto de parámetros de imagen. Una fila de mosaicos se define como una región rectangular de CTU que tiene una altura especificada por elementos de sintaxis en el conjunto de parámetros de imagen y una anchura igual a la anchura de la imagen. Una escaneo de mosaico se define como una
- 15 ordenación secuencial específica de las CTU que particionan una imagen en la que las CTU se ordenan consecutivamente en un escaneo de trama de CTU en un mosaico, mientras que los mosaicos de una imagen se ordenan consecutivamente en un escaneo de trama de los mosaicos de la imagen.

- 20 Esta sección especifica cómo se particiona una imagen en segmentos y mosaicos. Las imágenes se dividen en segmentos y mosaicos. Un segmento es una secuencia de mosaicos que cubren una región rectangular de una imagen. Un mosaico es una secuencia de CTU que cubren una región rectangular de una imagen.

- 25 Cuando una imagen se codifica usando tres planos de color separados ( $\text{separate\_colour\_plane\_flag}$  es igual a 1), un segmento contiene solo CTU de un componente de color que se identifica por el valor correspondiente de  $\text{colour\_plane\_id}$ , y cada matriz de componentes de color de una imagen incluye segmentos que tienen el mismo valor  $\text{colour\_plane\_id}$ . Los segmentos codificados con diferentes valores de  $\text{colour\_plane\_id}$  dentro de una imagen se pueden intercalar entre sí bajo la restricción de que, para cada valor de  $\text{colour\_plane\_id}$ , las unidades NAL del segmento codificado con ese valor de  $\text{colour\_plane\_id}$  deben estar en el orden de dirección
- 30 de CTU creciente en el orden de escaneo de mosaico para la primera CTU de cada unidad NAL de segmento codificado. Se debe tener en cuenta que cuando  $\text{separate\_colour\_plane\_flag}$  es igual a 0, cada CTU de una imagen está contenida en exactamente un segmento. Cuando  $\text{separate\_colour\_plane\_flag}$  es igual a 1, cada CTU de un componente de color está contenido en exactamente un segmento (por ejemplo, la información de cada CTU de una imagen está presente en exactamente tres segmentos y estos tres segmentos tienen diferentes valores de  $\text{colour\_plane\_id}$ ).

- 35 Las siguientes divisiones de elementos de procesamiento de esta memoria descriptiva forman particionamiento espacial o por componente: la división de cada imagen en componentes; la división de cada componente en CTB; la división de cada imagen en columnas de mosaicos; la división de cada imagen en filas de mosaicos; la división de cada columna de mosaicos en mosaicos; la división de cada fila de mosaicos en mosaicos; la
- 40 división de cada mosaico en CTU; la división de cada imagen en segmentos; la división de cada segmento en mosaicos; la división de cada segmento en CTU; la división de cada CTU en CTB; la división de cada CTB en bloques de codificación, excepto que los CTB estén incompletos en el límite del componente derecho cuando el ancho del componente no es un múltiplo entero del tamaño del CTB y los CTB estén incompletos en el límite del componente inferior cuando la altura del componente no es un múltiplo entero del tamaño del CTB; la
- 45 división de cada CTU en unidades de codificación, excepto que las CTU estén incompletas en el límite derecho de la imagen cuando el ancho de la imagen en las muestras de luma no es un múltiplo entero del tamaño del CTB de luma y las CTU estén incompletas en el límite inferior de la imagen cuando la altura de la imagen en las muestras de luma no es un múltiplo entero del tamaño del CTB de luma; la división de cada unidad de codificación en unidades de transformación; la división de cada unidad de codificación en bloques de
- 50 codificación; la división de cada bloque de codificación en bloques de transformación; y la división de cada unidad de transformación en bloques de transformación.

- Las entradas al proceso de derivación para la disponibilidad de bloques vecinos son la ubicación de luma ( $x_{\text{Curr}}, y_{\text{Curr}}$ ) de la muestra superior izquierda del bloque actual en relación con la muestra de luma superior
- 55 izquierda de la imagen actual, y la ubicación de luma ( $x_{\text{NbY}}, y_{\text{NbY}}$ ) cubierta por un bloque vecino en relación con la muestra de luma superior izquierda de la imagen actual. Los resultados de este proceso son la disponibilidad del bloque vecino que cubre la ubicación ( $x_{\text{NbY}}, y_{\text{NbY}}$ ), denotado como  $\text{availableN}$ . La disponibilidad del bloque vecino  $\text{availableN}$  se deriva de la siguiente manera. Si una o más de las siguientes condiciones son verdaderas,  $\text{availableN}$  se establece igual a falso. El  $\text{top\_left\_tile\_id}$  de segmento que contiene el bloque vecino difiere en valor del  $\text{top\_left\_tile\_id}$  del segmento que contiene el bloque actual o el bloque
- 60 vecino está contenido en un mosaico diferente al bloque actual.

- El proceso de escaneo de trama y mosaico CTB es el siguiente. La lista  $\text{ColWidth}[i]$  para  $i$  que va desde 0 a  $\text{num\_tile\_columns\_minus1}$ , inclusive, que especifica el ancho de la  $i$ -ésima columna de mosaico en unidades
- 65 de CTB, se deriva de la siguiente manera.

```

if( uniform_tile_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        ColWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) /
( num_tile_columns_minus1 + 1 ) --
                                ( i * PicWidthInCtbsY ) /
( num_tile_columns_minus1 + 1 )
else {
    ColWidth[ num_tile_columns_minus1 ] = PicWidthInCtbsY (6-1)
    for( i = 0; i < num_tile_columns_minus1; i++ ) {
        ColWidth[ i ] = tile_column_width_minus1[ i ] + 1
        ColWidth[ num_tile_columns_minus1 ] -= ColWidth[ i ]
    }
}

```

La lista RowHeight[ j ] para j que va desde 0 a num\_tile\_rows\_minus1, inclusive, que especifica la altura de la j-ésima fila de mosaicos en unidades de CTB, se deriva de la siguiente manera.

```

5  if( uniform_tile_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        RowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
) --
                                ( j * PicHeightInCtbsY ) / (
num_tile_rows_minus1 + 1 )
else {
    RowHeight[ num_tile_rows_minus1 ] = PicHeightInCtbsY (6-2)
    for( j = 0; j < num_tile_rows_minus1; j++ ) {
        RowHeight[ j ] = tile_row_height_minus1[ j ] + 1
        RowHeight[ num_tile_rows_minus1 ] -= RowHeight[ j ]
    }
}

```

La lista ColBd[ i ] para i que va de 0 a num\_tile\_columns\_minus1 + 1, inclusive, que especifica la ubicación del límite de columna de mosaico i-ésimo en unidades de CTB, se deriva de la siguiente manera: for( ColBd[ 0 ] = 0, i = 0; i <= num\_tile\_columns\_minus1; i++ )

```

10 ColBd[ i + 1 ] = ColBd[ i ] + ColWidth[ i ] (6-3)

```

La lista RowBd[ j ] para j que va de 0 a num\_tile\_rows\_minus1 + 1, inclusive, que especifica la ubicación del límite de fila de mosaico j-ésimo en unidades de CTB, se deriva de la siguiente manera:

```

15 for( RowBd[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
    RowBd[ j + 1 ] = RowBd[ j ] + RowHeight[ j ] (6-4)

```

La lista CtbAddrRsToTs[ ctbAddrRs ] para ctbAddrRs que va de 0 a PicSizeInCtbsY - 1, inclusive, que especifica la conversión de una dirección CTB en un escaneo de trama CTB de una imagen a una dirección CTB en un escaneo de mosaico, se deriva de la siguiente manera:

```

25 for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
    tbX = ctbAddrRs % PicWidthInCtbsY

```

```

tbY = ctbAddrRs / PicWidthInCtbsY
for( i = 0; i <= num_tile_columns_minus1; i++)
    if( tbX >= ColBd[ i ] )
        tileX = i
5  for( j = 0; j <= num_tile_rows_minus1; j++ ) (6-5)
    if( tbY >= RowBd[ j ] )
        tileY = j
    CtbAddrRsToTs[ ctbAddrRs ] = 0
    for( i = 0; i < tileX; i++ )
10  CtbAddrRsToTs[ ctbAddrRs ] += RowHeight[ tileY ] * ColWidth[ i ]
    for( j = 0; j < tileY; j++ )
        CtbAddrRsToTs[ ctbAddrRs ] += PicWidthInCtbsY * RowHeight[ j ]
    CtbAddrRsToTs[ ctbAddrRs ] += ( tbY - RowBd[ tileY ]
    ) * ColWidth[ tileX ] + tbX - ColBd[ tileX ]
15 }

```

La lista CtbAddrTsToRs[ctbAddrTs] para ctbAddrTs que va desde 0 hasta PicSizeInCtbsY - 1, inclusive, que especifica la conversión de una dirección CTB en escaneo de mosaico a una dirección CTB en escaneo de trama CTB de una imagen, se deriva de la siguiente manera:

```

20 for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) (6-6)

```

CtbAddrTsToRs[ CtbAddrRsToTs[ctbAddrRs] ] = ctbAddrRs La lista TileId[ctbAddrTs] para ctbAddrTs que va desde 0 hasta PicSizeInCtbsY - 1, inclusive, que especifica la conversión de una dirección CTB en escaneo de mosaico a un ID de mosaico, se deriva de la siguiente manera:

```

for( j = 0; tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )
        for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ ) (6-7)
30  for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
            TileId[ CtbAddrRsToTs[ y * PicWidthInCtbsY + x ] ] =
                explicit_tile_id_flag ? tile_id_val[ i ][ j ] :
tileIdx

```

35 La lista NumCtusInTile[ tileIdx ] para tileIdx que va de 0 a PicSizeInCtbsY - 1, inclusive, que especifica la conversión de un índice de mosaico al número de CTU en el mosaico, se deriva de la siguiente manera:

```

for( j = 0; tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ ) (6-8)
40  NumCtusInTile[ tileIdx ] = ColWidth[ i ] * RowHeight[ j ]

```

El conjunto TileIdToIdx[ tileId ] para un conjunto de valores NumTilesInPic tileId que especifican la conversión de un ID de mosaico a un índice de mosaico y la lista FirstCtbAddrTs[ tileIdx ] para tileIdx que va desde 0 a NumTilesInPic - 1, inclusive, que especifica la conversión de un ID de mosaico a la dirección CTB en el escaneo de mosaico del primer CTB en el mosaico se deriva de la siguiente manera:

```

for( ctbAddrTs = 0; tileIdx = 0; tileStartFlag = 1; ctbAddrTs < PicSizeInCtbsY; ctbAddrTs++ )
{
    if( tileStartFlag ) {
50  TileIdToIdx[ TileId[ ctbAddrTs ] ] = tileIdx
        FirstCtbAddrTs[ tileIdx ] = ctbAddrTs (6-9)
        tileStartFlag = 0
    }
    tileEndFlag = ctbAddrTs == PicSizeInCtbsY - 1 || TileId[ ctbAddrTs + 1 ] != TileId[ ctbAddrTs ]
55  if( tileEndFlag ) {
        tileIdx++
        tileStartFlag = 1
    }
}
60 }

```

Los valores de ColumnWidthInLumaSamples[ i ], que especifican el ancho de la i-ésima columna de mosaico en unidades de muestras de luma, se establecen igual a ColWidth[ i ] << CtbLog2SizeY para i, que va desde 0 hasta num\_tile\_columns\_minus1, inclusive. Los valores de RowHeightInLumaSamples[ j ], que especifican la altura de la fila de mosaico j en unidades de muestras de luma, se establecen igual a RowHeight[ j ] << CtbLog2SizeY para j, que va desde 0 hasta num\_tile\_rows\_minus1, inclusive.

La sintaxis RBSP del conjunto de parámetros de imagen es la siguiente:

Tabla 1

| pic_parameter_set_rbsp( ) {                        | Descriptor |
|----------------------------------------------------|------------|
| pps_pic_parameter_set_id                           | ue(v)      |
| pps_seq_parameter_set_id                           | ue(v)      |
| transform_skip_enabled_flag                        | u(1)       |
| single_tile_in_pic_flag                            | u(1)       |
| if( ! single_tile_in_pic_flag ) {                  |            |
| num_tile_columns_minus1                            | ue(v)      |
| num_tile_rows_minus1                               | ue(v)      |
| }                                                  |            |
| tile_id_len_minus1                                 | ue(v)      |
| explicit_tile_id_flag                              | u(1)       |
| if( explicit_tile_id_flag )                        |            |
| for( i = 0; i <= num_tile_columns_minus1; i++ )    |            |
| for( j = 0; j <= num_tile_rows_minus1; j++ )       |            |
| tile_id_val[ i ][ j ]                              | u(v)       |
| if( ! single_tile_in_pic_flag ) {                  |            |
| uniform_tile_spacing_flag                          | u(1)       |
| if( ! uniform_tile_spacing_flag ) {                |            |
| for( i = 0; i < num_tile_columns_minus1; i++ )     |            |
| tile_column_width_minus1[ i ]                      | ue(v)      |
| for( i = 0; i < num_tile_rows_minus1; i++ )        |            |
| tile_row_height_minus1[ i ]                        | ue(v)      |
| }                                                  |            |
| tile_boundary_treated_as_pic_boundary_flag         | u(1)       |
| if( ! tile_boundary_treated_as_pic_boundary_flag ) |            |
| loop_filter_across_tiles_enabled_flag              | u(1)       |
| }                                                  |            |

|                       |  |
|-----------------------|--|
| rbsp_trailing_bits( ) |  |
| }                     |  |

La sintaxis de la cabecera de segmento se cambia de la siguiente manera.

Tabla 2

5

| slice_header() {                                                                                                                                                                                                                                                                     | Descriptor |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| slice_pic_parameter_set_id                                                                                                                                                                                                                                                           | ue(v)      |
| single_tile_in_slice_flag // Misma nota que a continuación                                                                                                                                                                                                                           | u(1)       |
| top_left_tile_id // Tenga en cuenta que esto es necesario incluso cuando solo hay un mosaico en la imagen para permitir la extracción de un solo mosaico con restricciones de movimiento para que sea un flujo de bits conforme sin la necesidad de cambiar la cabecera de segmento. | u(v)       |
| if( ! single_tile_in_slice_flag )                                                                                                                                                                                                                                                    |            |
| bottom_right_tile_id                                                                                                                                                                                                                                                                 | u(v)       |
| slice_type                                                                                                                                                                                                                                                                           | ue(v)      |
| if ( slice_type != I )                                                                                                                                                                                                                                                               |            |
| log2_diff_ctu_max_bt_size                                                                                                                                                                                                                                                            | ue(v)      |
| dep_quant_enabled_flag                                                                                                                                                                                                                                                               | u(1)       |
| if( !dep_quant_enabled_flag )                                                                                                                                                                                                                                                        |            |
| sign_data_hiding_enabled_flag                                                                                                                                                                                                                                                        | u(1)       |
| if( !tile_boundary_treated_as_pic_boundary_flag )                                                                                                                                                                                                                                    |            |
| slice_boundary_treated_as_pic_boundary_flag                                                                                                                                                                                                                                          |            |
| if( ! single_tile_in_slice_flag ) {                                                                                                                                                                                                                                                  |            |
| offset_len_minus1                                                                                                                                                                                                                                                                    | ue(v)      |
| for( i = 0; i < NumTilesInSlice - 1; i++ )                                                                                                                                                                                                                                           |            |
| entry_point_offset_minus1[ i ]                                                                                                                                                                                                                                                       | u(v)       |
| }                                                                                                                                                                                                                                                                                    |            |
| byte_alignment( )                                                                                                                                                                                                                                                                    |            |
| }                                                                                                                                                                                                                                                                                    |            |

La sintaxis de slice\_data() es la siguiente:

Tabla 3

10

|                                                                                      |            |
|--------------------------------------------------------------------------------------|------------|
| slice_data( ) {                                                                      | Descriptor |
| tileIdx = TileIdToIdx[ top_left_tile_id ]                                            |            |
| for( j = 0; j < NumTileRowsInSlice; j++, tileIdx += num_tile_columns_minus1 + 1 ) {  |            |
| for( i = 0, CurrTileIdx = tileIdx; i < NumTileColumnsInSlice; i++, CurrTileIdx++ ) { |            |
| ctbAddrInTs = FirstCtbAddrTs[ CurrTileIdx ]                                          |            |
| for( k = 0; k < NumCtusInTile[ CurrTileIdx ]; k++, ctbAddrInTs++ ) {                 |            |
| CtbAddrInRs = CtbAddrTsToRs[ ctbAddrInTs ]                                           |            |
| coding_tree_unit( )                                                                  |            |
| }                                                                                    |            |
| end_of_tile_one_bit /* igual a 1 */                                                  | ae (v)     |
| if( i < NumTileRowsInSlice - 1    j < NumTileColumnsInSlice - 1                      |            |
| )                                                                                    |            |
| byte_alignment( )                                                                    |            |
| }                                                                                    |            |
| }                                                                                    |            |
| }                                                                                    |            |

La semántica Rbsp del conjunto de parámetros de imagen es la siguiente: el single\_tile\_in\_pic\_flag se establece igual a uno para especificar que solo hay un mosaico en cada imagen que hace referencia al PPS. El single\_picture\_in\_pic\_flag se establece igual a cero para especificar que hay más de un mosaico en el segmento que hace referencia al PPS. La conformidad del flujo de bits puede requerir que el valor de single\_tile\_in\_pic\_flag sea igual para todos los PPS que se activan dentro de una secuencia de vídeo codificada (CVS). El num\_tile\_columns\_minus1 más 1 especifica el número de columnas de mosaicos que particionan la imagen. El num\_tile\_columns\_minus1 debe estar en el intervalo de cero a PicWidthInCtbsY - 1, inclusive. Cuando no está presente, se infiere que el valor de num\_tile\_columns\_minus1 es igual a cero. El num\_tile\_rows\_minus1 más 1 especifica el número de filas de mosaicos que particionan la imagen. El num\_tile\_rows\_minus1 debe estar en el intervalo de cero a PicHeightInCtbsY - 1, inclusive. Cuando no está presente, se infiere que el valor de num\_tile\_rows\_minus1 es igual a cero. La variable NumTilesInPic se establece igual a (num\_tile\_columns\_minus1 + 1) \* (num\_tile\_rows\_minus1 + 1).

- 15 Cuando single\_tile\_in\_pic\_flag es igual a cero, NumTilesInPic deberá ser mayor que cero. El tile\_id\_len\_minus1 más 1 especifica el número de bits usados para representar el elemento de sintaxis tile\_id\_val[ i ][ j ], cuando está presente, en el PPS y los elementos de sintaxis top\_left\_tile\_id y bottom\_right\_tile\_id, cuando están presentes, en las cabeceras de segmento que hacen referencia al PPS. El valor de tile\_id\_len\_minus1 debe estar en el intervalo de Ceil( Log2( NumTilesInPic ) ) a 15, inclusive. El explicit\_tile\_id\_flag se establece
- 20 igual a uno para especificar que el ID de mosaico para cada mosaico se señala explícitamente. El explicit\_tile\_id\_flag se establece igual a cero para especificar que los ID de mosaico no se señalizan explícitamente. El tile\_id\_val[ i ][ j ] especifica el ID de mosaico del mosaico de la i-ésima columna de mosaicos y de la j-ésima fila de mosaicos. La longitud de tile\_id\_val[ i ][ j ] es tile\_id\_len\_minus1 + 1 bits.
- 25 Para cualquier entero m en el intervalo de cero a num\_tile\_columns\_minus1, inclusive, y cualquier entero n en el intervalo de cero a num\_tile\_rows\_minus1, inclusive, tile\_id\_val[ i ][ j ] no será igual a tile\_id\_val[ m ][ n ] cuando i no sea igual a m o j no sea igual a n, y tile\_id\_val[ i ][ j ] será menor que tile\_id\_val[ m ][ n ] cuando j \* ( num\_tile\_columns\_minus1 + 1 ) + i sea menor que n \* ( num\_tile\_columns\_minus1 + 1 ) + m. El uniform\_tile\_spacing\_flag se establece igual uno para especificar que los límites de las columnas de mosaicos y, asimismo, los límites de las filas de mosaicos se distribuyen uniformemente en la imagen. El
- 30 uniform\_tile\_spacing\_flag se establece igual a cero para especificar que los límites de las columnas de

mosaicos, y asimismo los límites de las filas de mosaicos, no se distribuyen uniformemente en la imagen, sino que se señalizan explícitamente usando los elementos de sintaxis `tile_column_width_minus1[ i ]` y `tile_row_height_minus1[ i ]`. Cuando no está presente, se infiere que el valor de `uniform_tile_spacing_flag` es igual a 1. El `tile_column_width_minus1[ i ]` más 1 especifica el ancho de la *i*-ésima columna de mosaico en unidades de CTB. El `tile_row_height_minus1[ i ]` más 1 especifica la altura de la fila de mosaicos *i*-ésima en unidades de CTB.

Las siguientes variables se derivan invocando el proceso de conversión de escaneo de mosaico y de trama CTB: la lista `ColWidth[ i ]` para *i* que va de 0 a `num_tile_columns_minus1`, inclusive, que especifica el ancho de la columna de mosaico *i*-ésima en unidades de CTB; la lista `RowHeight[ j ]` para *j* que va de 0 a `num_tile_rows_minus1`, inclusive, que especifica la altura de la fila de mosaico *j*-ésima en unidades de CTB; la lista `ColBd[ i ]` para *i* que va de 0 a `num_tile_columns_minus1 + 1`, inclusive, que especifica la ubicación del límite de la columna de mosaico *i*-ésima en unidades de CTB; la lista `RowBd[ j ]` para *j* que va de 0 a `num_tile_rows_minus1 + 1`, inclusive, que especifica la ubicación del límite de la fila de mosaico *j*-ésima en unidades de CTB; la lista `CtbAddrRsToTs[ctbAddrRs]` para `ctbAddrRs` que va desde 0 hasta `PicSizeInCtbsY - 1`, inclusive, que especifica la conversión de una dirección CTB en el escaneo de trama CTB de una imagen a una dirección CTB en el escaneo de mosaico; la lista `CtbAddrTsToRs[ctbAddrTs]` para `ctbAddrTs` que va desde 0 hasta `PicSizeInCtbsY - 1`, inclusive, que especifica la conversión de una dirección CTB en el escaneo de mosaico a una dirección CTB en el escaneo de trama CTB de una imagen; la lista `TileId[ctbAddrTs]` para `ctbAddrTs` que va desde 0 hasta `PicSizeInCtbsY - 1`, inclusive, que especifica la conversión de una dirección CTB en el escaneo de mosaico a un ID de mosaico; la lista `NumCtusInTile[ tileIdx ]` para `tileIdx` que va de 0 a `PicSizeInCtbsY - 1`, inclusive, que especifica la conversión de un índice de mosaico al número de CTU en el mosaico; el conjunto `TileIdToIdx[ tileId ]` para un conjunto de valores `tileId` `NumTilesInPic` que especifican la conversión de un ID de mosaico a un índice de mosaico y la lista `FirstCtbAddrTs[ tileIdx ]` para `tileIdx` que va de 0 a `NumTilesInPic - 1`, inclusive, que especifica la conversión de un ID de mosaico a la dirección CTB en el escaneo de mosaico del primer CTB en el mosaico; las listas `ColumnWidthInLumaSamples[ i ]` para *i* que va de 0 a `num_tile_columns_minus1`, inclusive, que especifica el ancho de la *i*-ésima columna de mosaico en unidades de muestras de luma; y la lista `RowHeightInLumaSamples[ j ]` para *j* que va desde 0 a `num_tile_rows_minus1`, inclusive, que especifica la altura de la *j*-ésima fila de mosaicos en unidades de muestras de luma.

Los valores de `ColumnWidthInLumaSamples[ i ]` para *i* que va desde 0 hasta `num_tile_columns_minus1`, inclusive, y `RowHeightInLumaSamples[ j ]` para *j* que va desde 0 hasta `num_tile_rows_minus1`, inclusive, deben ser todos mayores que 0. El `tile_boundary_treated_as_picture_boundary_flag` se establece igual a uno para especificar que cada límite de mosaico se trata igual que el límite de imagen en el proceso de decodificación para imágenes que hacen referencia al PPS. El `tile_boundary_treated_as_picture_boundary_flag` se establece igual a cero para especificar que cada límite de mosaico puede o no ser tratado igual que el límite de imagen en el proceso de decodificación de imágenes que hacen referencia al PPS. Cuando no está presente, se infiere que el valor de `tile_boundary_treated_as_picture_boundary_flag` es igual a uno. El `loop_filter_across_tiles_enabled_flag` se establece igual a uno para especificar que se pueden realizar operaciones de filtrado en bucle en los límites de mosaico en imágenes que hacen referencia al PPS. El `loop_filter_across_tiles_enabled_flag` se establece igual a cero para especificar que no se pueden realizar operaciones de filtrado en bucle en los límites de mosaico en imágenes que hacen referencia al PPS. Las operaciones de filtrado en bucle incluyen el filtro de desbloqueo, el filtro de desplazamiento adaptativo de muestra y las operaciones de filtro de bucle adaptativo. Cuando no está presente, se infiere que el valor de `loop_filter_across_tiles_enabled_flag` es igual a cero.

La semántica de la cabecera de segmento es la siguiente. Cuando esté presente, el valor del elemento de sintaxis de la cabecera de segmento `slice_pic_parameter_set_id` deberá ser igual en todas las cabeceras de segmento de una imagen codificada. El `slice_pic_parameter_set_id` especifica el valor de `pps_pic_parameter_set_id` para el PPS en uso. El valor de `slice_pic_parameter_set_id` debe estar en el intervalo de 0 a 63, inclusive. El `single_tile_in_slice_flag` se establece igual a uno para especificar que solo hay un mosaico en el segmento. El `single_picture_in_pic_flag` se establece igual a cero para especificar que hay más de un mosaico en el segmento. El `top_left_tile_id` especifica el ID de mosaico del mosaico ubicado en la esquina superior izquierda del segmento. La longitud de `top_left_tile_id` es `tile_id_len_minus1 + 1` bits. El valor de `top_left_tile_id` no debe ser igual al valor de `top_left_tile_id` de ninguna otra unidad NAL de segmento codificado de la misma imagen codificada. Cuando hay más de un segmento en una imagen, el orden de decodificación de los segmentos en la imagen deberá ser en valor creciente de `top_left_tile_id`. El `bottom_right_tile_id` especifica el ID de mosaico ubicado en la esquina inferior derecha del segmento. La longitud de `bottom_right_tile_id` es `tile_id_len_minus1 + 1` bits. Cuando no está presente, se infiere que el valor de `bottom_right_tile_id` es igual a `top_left_tile_id`.

Las variables `NumTileRowsInSlice`, `NumTileColumnsInSlice` y `NumTilesInSlice` se derivan de la siguiente manera:

$$\text{deltaTileIdx} = \text{TileIdToIdx}[\text{bottom\_right\_tile\_id}] - \text{TileIdToIdx}[\text{top\_left\_tile\_id}]$$

$$\text{NumTileRowsInSlice} = ( \text{deltaTileIdx} / ( \text{num\_tile\_columns\_minus1} + 1 ) ) + 1 \quad (7-25)$$

$$\text{NumTileColumnsInSlice} = ( \text{deltaTileIdx} \% ( \text{num\_tile\_columns\_minus1} + 1 ) ) + 1$$

$$\text{NumTilesInSlice} = \text{NumTileRowsInSlice} * \text{NumTileColumnsInSlice}$$

El slice\_type especifica el tipo de codificación del segmento según la tabla 4.

Tabla 4

| slice_type | Nombre de slice_type |
|------------|----------------------|
| 0          | B (segmento B)       |
| 1          | P (segmento P)       |
| 2          | I (segmento I)       |

Cuando nal\_unit\_type tiene un valor en el intervalo de TBD, inclusive, por ejemplo, la imagen es una Imagen de Acceso IntraAleatorio (IRAP), slice\_type deberá ser igual a dos. El log2\_diff\_ctu\_max\_bt\_size especifica la diferencia entre el tamaño de CTB de luma y el tamaño máximo de luma (ancho o alto) de un bloque de codificación que se puede dividir usando una división binaria. El valor de log2\_diff\_ctu\_max\_bt\_size debe estar en el intervalo de cero a CtbLog2SizeY - MinCbLog2SizeY, inclusive. Cuando log2\_diff\_ctu\_max\_bt\_size no está presente, se infiere que el valor de log2\_diff\_ctu\_max\_bt\_size es igual a dos.

Las variables MinQtLog2SizeY, MaxBtLog2SizeY, MinBtLog2SizeY, MaxTtLog2SizeY, MinTtLog2SizeY, MaxBtSizeY, MinBtSizeY, MaxTtSizeY, MinTtSizeY y MaxMttDepth se derivan de la siguiente manera:

$$\text{MinQtLog2SizeY} = (\text{slice\_type} == 1) ? \text{MinQtLog2SizeIntraY} : \text{MinQtLog2SizeInterY} \quad (7-26)$$

$$\text{MaxBtLog2SizeY} = \text{CtbLog2SizeY} - \text{log2\_diff\_ctu\_max\_bt\_size} \quad (7-27)$$

$$\text{MinBtLog2SizeY} = \text{MinCbLog2SizeY} \quad (7-28)$$

$$\text{MaxTtLog2SizeY} = (\text{slice\_type} == 1) ? 5 : 6 \quad (7-29)$$

$$\text{MinTtLog2SizeY} = \text{MinCbLog2SizeY} \quad (7-30)$$

$$\text{MinQtSizeY} = 1 \ll \text{MinQtLog2SizeY} \quad (7-31)$$

$$\text{MaxBtSizeY} = 1 \ll \text{MaxBtLog2SizeY} \quad (7-32)$$

$$\text{MinBtSizeY} = 1 \ll \text{MinBtLog2SizeY} \quad (7-33)$$

$$\text{MaxTtSizeY} = 1 \ll \text{MaxTtLog2SizeY} \quad (7-34)$$

$$\text{MinTtSizeY} = 1 \ll \text{MinTtLog2SizeY} \quad (7-35)$$

$$\text{MaxMttDepth} = (\text{slice\_type} == 1) ? \text{max\_mtt\_hierarchy\_depth\_intra\_slices} : \text{max\_mtt\_hierarchy\_depth\_inter\_slices} \quad (7-36)$$

El dep\_quant\_enabled se establece igual a cero para especificar que la cuantificación dependiente está deshabilitada. El dep\_quant\_enabled\_flag se establece igual a uno para especificar que la cuantificación dependiente está habilitada. El sign\_data\_hiding\_enabled\_flag se establece igual a cero para especificar que la ocultación del bit de signo está deshabilitada. El sign\_data\_hiding\_enabled\_flag se establece igual a uno para especificar que la ocultación del bit de signo está habilitada. Cuando sign\_data\_hiding\_enabled\_flag no está presente, se infiere que es igual a cero. El slice\_boundary\_treated\_as\_pic\_boundary\_flag se establece igual a uno para especificar que cada límite de segmento se trata igual que límite de imagen en el proceso de decodificación. El slice\_boundary\_treated\_as\_pic\_boundary\_flag igual a cero especifica que cada límite de mosaico puede o no ser tratado igual que el límite de imagen en el proceso de decodificación. Cuando no está presente, se infiere que el valor de slice\_boundary\_treated\_as\_pic\_boundary\_flag es igual a uno. El offset\_len\_minus1 más 1 especifica la longitud, en bits, de los elementos de sintaxis entry\_point\_offset\_minus1[i]. El valor de offset\_len\_minus1 deberá estar en el intervalo de 0 a 31, inclusive. El entry\_point\_offset\_minus1[i] más 1 especifica el desplazamiento del punto de entrada i-ésimo en bytes y está representado por offset\_len\_minus1 más 1 bits. Los datos de segmento que siguen a la cabecera de segmento constan de subconjuntos NumTilesInSlice, con valores de índice de subconjunto que va desde 0 a NumTilesInSlice - 1, inclusive. El primer byte de los datos de segmento se considera el byte cero. Cuando están presentes, los bytes de prevención de emulación que aparecen en la parte de datos de segmento de la unidad NAL de segmento codificado se cuentan como parte de los datos de segmento con fines de identificación de subconjunto.

El subconjunto cero incluye los bytes cero a entry\_point\_offset\_minus1[0], inclusive, de los datos de segmento de segmento codificados, el subconjunto k, con k en el intervalo de 1 a NumTilesInSlice - 2, inclusive, incluye

los bytes firstByte[ k ] a lastByte[ k ], inclusive, de los datos de segmento codificados con firstByte[ k ] y lastByte[ k ] definidos como:

$$\text{firstByte}[k] = \sum_{a=1}^k (\text{entry\_point\_offset\_minus1}[a-1] + 1) \quad (7-37)$$

5

$$\text{lastByte}[k] = \text{firstByte}[k] + \text{entry\_point\_offset\_minus1}[k] \quad (7-38)$$

El último subconjunto (con índice de subconjunto igual a NumTilesInSlice - 1) incluye los bytes restantes de los datos de segmento codificados. Cada subconjunto incluirá todos los bits codificados de todas las CTU en el segmento que estén dentro del mismo mosaico.

La semántica general de los datos de segmento es la siguiente: el end\_of\_tile\_one\_bit debe ser igual a uno. Para cada mosaico, las variables LeftBoundaryPos, TopBoundaryPos, RightBoundaryPos y BotBoundaryPos se derivan de la siguiente manera. Si tile\_boundary\_treated\_as\_pic\_boundary\_flag es igual a verdadero, se aplica lo siguiente:

$$\begin{aligned} \text{tileRowIdx} &= \text{CurrTileIdx} \% (\text{num\_tile\_columns\_minus1} + 1) & (7-40) \\ \text{LeftBoundaryPos} &= \text{ColBd}[\text{tileColIdx}] << \text{CtbLog2SizeY} & (7-41) \\ \text{RightBoundaryPos} &= ((\text{ColBd}[\text{tileColIdx}] + \text{ColWidth}[\text{tileColIdx}]) << \text{CtbLog2SizeY}) - 1 & (7-42) \\ \text{TopBoundaryPos} &= \text{RowBd}[\text{tileRowIdx}] << \text{CtbLog2SizeY} & (7-43) \\ \text{BotBoundaryPos} &= ((\text{RowBd}[\text{tileRowIdx}] + \text{RowHeight}[\text{tileRowIdx}]) << \text{CtbLog2SizeY}) - 1 & (7-44) \end{aligned}$$

De lo contrario, si slice\_boundary\_treated\_as\_pic\_boundary\_flag es igual a verdadero, se aplica lo siguiente:

$$\begin{aligned} \text{sliceFirstTileColIdx} &= \text{TileIdToIdx}[\text{top\_left\_tile\_id}] / (\text{num\_tile\_columns\_minus1} + 1) & (7-45) \\ \text{sliceFirstTileRowIdx} &= \text{TileIdToIdx}[\text{top\_left\_tile\_id}] \% (\text{num\_tile\_columns\_minus1} + 1) & (7-46) \\ \text{sliceLastTileColIdx} &= \text{TileIdToIdx}[\text{bottom\_right\_tile\_id}] / (\text{num\_tile\_columns\_minus1} + 1) & (7-47) \\ \text{sliceLastTileRowIdx} &= \text{TileIdToIdx}[\text{bottom\_right\_tile\_id}] \% (\text{num\_tile\_columns\_minus1} + 1) & (7-48) \\ \text{LeftBoundaryPos} &= \text{ColBd}[\text{sliceFirstTileColIdx}] << \text{CtbLog2SizeY} & (7-49) \\ \text{RightBoundaryPos} &= ((\text{ColBd}[\text{sliceLastTileColIdx}] + \text{ColWidth}[\text{sliceLastTileColIdx}]) << \text{CtbLog2SizeY}) - 1 & (7-50) \\ \text{TopBoundaryPos} &= \text{RowBd}[\text{sliceFirstTileRowIdx}] << \text{CtbLog2SizeY} & (7-51) \\ \text{BotBoundaryPos} &= ((\text{RowBd}[\text{sliceLastTileRowIdx}] + \text{RowHeight}[\text{sliceLastTileRowIdx}]) << \text{CtbLog2SizeY}) - 1 & (7-52) \end{aligned}$$

De lo contrario (slice\_boundary\_treated\_as\_pic\_boundary\_flag es igual a FALSO), se aplica lo siguiente:

$$\begin{aligned} \text{LeftBoundaryPos} &= 0 & (7-53) \\ \text{RightBoundaryPos} &= \text{pic\_width\_in\_luma\_samples} - 1 & (7-54) \\ \text{TopBoundaryPos} &= 0 & (7-55) \\ \text{BotBoundaryPos} &= \text{pic\_height\_in\_luma\_samples} - 1 & (7-56) \end{aligned}$$

El proceso de derivación para la predicción del vector de movimiento de luma temporal es el siguiente. Si yCb >> CtbLog2SizeY es igual a yColBr >> CtbLog2SizeY, yColBr es menor que pic\_height\_in\_luma\_samples y xColBr es menor que pic\_width\_in\_luma\_samples, se aplica lo siguiente:

La variable colCb especifica el bloque de codificación de luma que cubre la ubicación modificada dada por ( ( xColBr >> 3 ) << 3, ( yColBr >> 3 ) << 3 ) dentro de la imagen coubicada especificada por ColPic. La ubicación de luma (xColCb, yColCb) se establece igual a la muestra superior izquierda del bloque de codificación de luma coubicado especificado por colCb en relación con la muestra de luma superior izquierda de la imagen coubicada especificada por ColPic. El proceso de derivación de vectores de movimiento coubicados se invoca con currCb, colCb, ( xColCb, yColCb ), reldxLX y el parámetro de control controlParaFlag se establece igual a 0 como entradas, y la salida se asigna a mvLXCol y availableFlagLXCol. Si yCb >> CtbLog2SizeY es igual a yColBr >> CtbLog2SizeY, yColBr es menor o igual que BotBoundaryPos y xColBr es menor o igual que RightBoundaryPos, se aplica lo siguiente. La variable colCb especifica el bloque de codificación de luma que cubre la ubicación modificada dada por ( ( xColBr >> 3 ) << 3, ( yColBr >> 3 ) << 3 ) dentro de la imagen coubicada especificada por ColPic. La ubicación de luma (xColCb, yColCb) se establece igual a la muestra superior izquierda del bloque de codificación de luma coubicado especificado por colCb en relación con la muestra de luma superior izquierda de la imagen coubicada especificada por ColPic. El proceso de derivación de vectores de movimiento coubicados se invoca con currCb, colCb, ( xColCb, yColCb ), reldxLX y el parámetro de control controlParaFlag se establece igual a 0 como entradas, y la salida se asigna a mvLXCol y availableFlagLXCol.

En algunos ejemplos, el proceso de derivación para la predicción del vector de movimiento de luma temporal es el siguiente. Si yCb >> CtbLog2SizeY es igual a yColBr >> CtbLog2SizeY, yColBr es menor que pic\_height\_in\_luma\_samples y xColBr es menor que pic\_width\_in\_luma\_samples, se aplica lo siguiente. La

variable colCb especifica el bloque de codificación de luma que cubre la ubicación modificada dada por ( ( xColBr >> 3 ) << 3, ( yColBr >> 3 ) << 3 ) dentro de la imagen coubicada especificada por ColPic. La ubicación de luma (xColCb, yColCb) se establece igual a la muestra superior izquierda del bloque de codificación de luma coubicado especificado por colCb en relación con la muestra de luma superior izquierda de la imagen coubicada especificada por ColPic. El proceso de derivación de vectores de movimiento coubicados se invoca con currCb, colCb, ( xColCb, yColCb ), refIdxLX y el parámetro de control controlParaFlag se establece igual a 0 como entradas, y la salida se asigna a mvLXCol y availableFlagLXCol. Si yCb >> CtbLog2SizeY es igual a yColBr >> CtbLog2SizeY, se aplica lo siguiente. xColCtr = Min( xColCtr, RightBoundaryPos ) y yColCtr = Min( yColCtr, BotBoundaryPos ). La variable colCb especifica el bloque de codificación de luma que cubre la ubicación modificada dada por ( ( xColBr >> 3 ) << 3, ( yColBr >> 3 ) << 3 ) dentro de la imagen coubicada especificada por ColPic. La ubicación de luma (xColCb, yColCb) se establece igual a la muestra superior izquierda del bloque de codificación de luma coubicado especificado por colCb en relación con la muestra de luma superior izquierda de la imagen coubicada especificada por ColPic. El proceso de derivación de vectores de movimiento coubicados se invoca con currCb, colCb, ( xColCb, yColCb ), refIdxLX y el parámetro de control controlParaFlag se establece igual a 0 como entradas, y la salida se asigna a mvLXCol y availableFlagLXCol.

El proceso de interpolación de muestras de luma es el siguiente. Las entradas para este proceso son: una ubicación de luma en unidades de muestra completa (xIntL, yIntL); una ubicación de luma en unidades de muestra fraccionaria (xFracL, yFracL); y la matriz de muestra de referencia de luma refPicLXL. El resultado de este proceso es un valor de muestra de luma predicho predSampleLXL. Las variables shift1, shift2, shift3 se derivan de la siguiente manera. La variable shift1 se establece igual a Min( 4, BitDepthY - 8 ) y la variable shift2 se establece igual a 6 y la variable shift3 se establece igual a Max( 2, 14 - BitDepthY ). El valor de muestra de luma predicho predSampleLXL se deriva de la siguiente manera. Si tanto xFracL como yFracL son iguales a 0, el valor de predSampleLXL se deriva de la siguiente manera:

25 
$$\text{predSampleLXL} = \text{refPicLXL}[ \text{xIntL} ][ \text{yIntL} ] \ll \text{shift3} \quad (8-228)$$
  
De lo contrario, si xFracL no es igual a 0 e yFracL es igual a 0, el valor de predSampleLXL se deriva de la siguiente manera:

30 
$$\begin{aligned} & ( \text{fL}[ \text{xFracL}, 0 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} - 3 ) ][ \text{yIntL} ] + \\ & \text{fL}[ \text{xFracL} ][ 1 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} - 2 ) ][ \text{yIntL} ] + \\ & \text{fL}[ \text{xFracL} ][ 2 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} - 1 ) ][ \text{yIntL} ] + \\ & \text{fL}[ \text{xFracL} ][ 3 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} ) ][ \text{yIntL} ] + \\ & \text{fL}[ \text{xFracL} ][ 4 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 1 ) ][ \text{yIntL} ] + (8-228) \\ 35 & \text{fL}[ \text{xFracL} ][ 5 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 2 ) ][ \text{yIntL} ] + \\ & \text{fL}[ \text{xFracL} ][ 6 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 3 ) ][ \text{yIntL} ] + \\ & \text{fL}[ \text{xFracL} ][ 7 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 4 ) ][ \text{yIntL} ] ) \gg \text{shift1} \end{aligned}$$

De lo contrario, si xFracL es igual a 0 e yFracL no es igual a 0, el valor de predSampleLXL se deriva de la siguiente manera:

40 
$$\begin{aligned} & ( \text{fL}[ \text{yFracL}, 0 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} - 3 ) ] + \\ & \text{fL}[ \text{yFracL} ][ 1 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} - 2 ) ] + \\ & \text{fL}[ \text{yFracL} ][ 2 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} - 1 ) ] + \\ 45 & \text{fL}[ \text{yFracL} ][ 3 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} ) ] + \\ & \text{fL}[ \text{yFracL} ][ 4 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} + 1 ) ] + (8-228) \\ & \text{fL}[ \text{yFracL} ][ 5 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} + 2 ) ] + \\ & \text{fL}[ \text{yFracL} ][ 6 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} + 3 ) ] + \\ & \text{fL}[ \text{yFracL} ][ 7 ] * \text{refPicLXL}[ \text{xIntL} ][ \text{Clip3}( \text{TopBoundaryPos}, \text{BotBoundaryPos}, \text{yIntL} + 4 ) ] ) \gg \text{shift1} \end{aligned}$$

50 De lo contrario, si xFracL no es igual a 0 e yFracL no es igual a 0, el valor de predSampleLXL se deriva de la siguiente manera. La matriz de muestra temp[ n ] con n = 0..7, se deriva de la siguiente manera:

$$\text{temp}[ n ] =$$
  
55 
$$\begin{aligned} & ( \text{fL}[ \text{xFracL}, 0 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} - 3 ) ][ \text{yPosL} ] + \\ & \text{fL}[ \text{xFracL} ][ 1 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} - 2 ) ][ \text{yPosL} ] + \\ & \text{fL}[ \text{xFracL} ][ 2 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} - 1 ) ][ \text{yPosL} ] + \\ & \text{fL}[ \text{xFracL} ][ 3 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} ) ][ \text{yPosL} ] + \\ & \text{fL}[ \text{xFracL} ][ 4 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 1 ) ][ \text{yPosL} ] + (8-228) \\ 60 & \text{fL}[ \text{xFracL} ][ 5 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 2 ) ][ \text{yPosL} ] + \\ & \text{fL}[ \text{xFracL} ][ 6 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 3 ) ][ \text{yPosL} ] + \\ & \text{fL}[ \text{xFracL} ][ 7 ] * \text{refPicLXL}[ \text{Clip3}( \text{LeftBoundaryPos}, \text{RightBoundaryPos}, \text{xIntL} + 4 ) ][ \text{yPosL} ] ) \gg \text{shift1} \end{aligned}$$

El valor de muestra de luma predicho predSampleLXL se deriva de la siguiente manera:

65 
$$\text{predSampleLXL} = \text{fL}[ \text{yFracL} ][ 0 ] * \text{temp}[ 0 ] +$$

5  
10

```

fL[ yFracL ][ 1 ] * temp[ 1 ] +
fL[ yFracL ][ 2 ] * temp[ 2 ] +
fL[ yFracL ][ 3 ] * temp[ 3 ] +
fL[ yFracL ][ 4 ] * temp[ 4 ] + (8-228)
fL[ yFracL ][ 5 ] * temp[ 5 ] +
fL[ yFracL ][ 6 ] * temp[ 6 ] +
fL[ yFracL ][ 7 ] * temp[ 7 ] ) >> shift2
    
```

Tabla 5

| Posición de muestra fraccionaria p | Coeficientes del filtro de interpolación |              |              |              |              |              |              |              |
|------------------------------------|------------------------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|                                    | fL[ p ][ 0 ]                             | fL[ p ][ 1 ] | fL[ p ][ 2 ] | fL[ p ][ 3 ] | fL[ p ][ 4 ] | fL[ p ][ 5 ] | fL[ p ][ 6 ] | fL[ p ][ 7 ] |
| 1                                  | 0                                        | 1            | -3           | 63           | 4            | -2           | 1            | 0            |
| 2                                  | -1                                       | 2            | -5           | 62           | 8            | -3           | 1            | 0            |
| 3                                  | -1                                       | 3            | -8           | 60           | 13           | -4           | 1            | 0            |
| 4                                  | -1                                       | 4            | -10          | 58           | 17           | -5           | 1            | 0            |
| 5                                  | -1                                       | 4            | -11          | 52           | 26           | -8           | 3            | -1           |
| 6                                  | -1                                       | 3            | -9           | 47           | 31           | -10          | 4            | -1           |
| 7                                  | -1                                       | 4            | -11          | 45           | 34           | -10          | 4            | -1           |
| 8                                  | -1                                       | 4            | -11          | 40           | 40           | -11          | 4            | -1           |
| 9                                  | -1                                       | 4            | -10          | 34           | 45           | -11          | 4            | -1           |
| 10                                 | -1                                       | 4            | -10          | 31           | 47           | -9           | 3            | -1           |
| 11                                 | -1                                       | 3            | -8           | 26           | 52           | -11          | 4            | -1           |
| 12                                 | 0                                        | 1            | -5           | 17           | 58           | -10          | 4            | -1           |
| 13                                 | 0                                        | 1            | -4           | 13           | 60           | -8           | 3            | -1           |
| 14                                 | 0                                        | 1            | -3           | 8            | 62           | -5           | 2            | -1           |
| 15                                 | 0                                        | 1            | -2           | 4            | 63           | -3           | 1            | 0            |

El proceso de interpolación de muestras de croma es el siguiente. Las entradas para este proceso son: una ubicación de croma en unidades de muestra completa (xIntC, yIntC); una ubicación de croma en unidades de muestra fraccionaria de octavo (xFracC, yFracC); y la matriz de muestra de referencia de croma refPicLXC. El resultado de este proceso es un valor de muestra de croma predicho predSampleLXC. Las variables shift1, shift2, shift3, picWC y picHC se derivan de la siguiente manera. La variable shift1 se establece igual a Min( 4, BitDepthC - 8 ), y la variable shift2 se establece igual a 6 y la variable shift3 se establece igual a Max( 2, 14 - BitDepthC ). Las variables lPos, rPos, tPos y bPos se establecen de la siguiente manera:

lPos = LeftBoundaryPos / SubWidthC (8-228)

rPos = ( RightBoundaryPos + 1 ) / SubWidthC (8-228)

tPos = TopBoundaryPos / SubHeightC (8-228)

bPos = ( BotBoundaryPos + 1 ) / SubHeightC (8-228)

El valor de muestra de croma predicho predSampleLXC se deriva de la siguiente manera. Si tanto xFracC como yFracC son iguales a 0, el valor de predSampleLXC se deriva de la siguiente manera:

$$5 \quad \text{predSampleLXC} = \text{refPicLXC}[\text{xIntC}][\text{yIntC}] \ll \text{shift3} \quad (8-228)$$

De lo contrario, si xFracC no es igual a 0 e yFracC es igual a 0, el valor de predSampleLXC se deriva de la siguiente manera:

$$10 \quad \begin{aligned} & \text{fC}[\text{xFracC}][1] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC})][\text{yIntC}] + \\ & \text{fC}[\text{xFracC}][2] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC} + 1)][\text{yIntC}] + \\ & \text{fC}[\text{xFracC}][3] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC} + 2)][\text{yIntC}] \gg \text{shift1} \end{aligned} \quad (8-228)$$

De lo contrario, si xFracC es igual a 0 e yFracC no es igual a 0, el valor de predSampleLXC se deriva de la siguiente manera:

$$20 \quad \begin{aligned} & \text{fC}[\text{yFracC}][1] * \text{refPicLXC}[\text{xIntC}][\text{Clip3}(\text{tPos}, \text{bPos}, \text{yIntC})] + \\ & \text{fC}[\text{yFracC}][2] * \text{refPicLXC}[\text{xIntC}][\text{Clip3}(\text{tPos}, \text{bPos}, \text{yIntC} + 1)] + \\ & \text{fC}[\text{yFracC}][3] * \text{refPicLXC}[\text{xIntC}][\text{Clip3}(\text{tPos}, \text{bPos}, \text{yIntC} + 2)] \gg \text{shift1} \end{aligned} \quad (8-228)$$

De lo contrario, si xFracC no es igual a 0 y yFracC no es igual a 0, el valor de predSampleLXC es derivado de la siguiente manera. La matriz de muestra temp[n] con n = 0..3, se deriva de la siguiente manera:

$$25 \quad \begin{aligned} & \text{yPosC} = \text{Clip3}(\text{tPos}, \text{bPos}, \text{yIntC} + \text{n} - 1) \quad (8-228) \\ & \text{temp}[\text{n}] = (\text{fC}[\text{xFracC}][0] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC} - 1)][\text{yPosC}] + \\ & \quad \text{fC}[\text{xFracC}][1] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC})][\text{yPosC}] + \\ & \quad \text{fC}[\text{xFracC}][2] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC} + 1)][\text{yPosC}] + \\ & \quad \text{fC}[\text{xFracC}][3] * \text{refPicLXC}[\text{Clip3}(\text{IPos}, \text{rPos}, \text{xIntC} + 2)][\text{yPosC}]) \gg \text{shift1} \end{aligned} \quad (8-228)$$

30 El valor de muestra de croma predicho predSampleLXC se deriva de la siguiente manera:

$$35 \quad \begin{aligned} & \text{predSampleLXC} = (\text{fC}[\text{yFracC}][0] * \text{temp}[0] + \\ & \quad \text{fC}[\text{yFracC}][1] * \text{temp}[1] + \\ & \quad \text{fC}[\text{yFracC}][2] * \text{temp}[2] + \\ & \quad \text{fC}[\text{yFracC}][3] * \text{temp}[3]) \gg \text{shift2} \end{aligned} \quad (8-228)$$

Tabla 6

| Posición de muestra fraccionaria p | coeficientes del filtro de interpolación |              |              |              |
|------------------------------------|------------------------------------------|--------------|--------------|--------------|
|                                    | fC[ p ][ 0 ]                             | fC[ p ][ 1 ] | fC[ p ][ 2 ] | fC[ p ][ 3 ] |
| 1                                  | -1                                       | 63           | 2            | 0            |
| 2                                  | -2                                       | 62           | 4            | 0            |
| 3                                  | -2                                       | 60           | 7            | -1           |
| 4                                  | -2                                       | 58           | 10           | -2           |
| 5                                  | -3                                       | 57           | 12           | -2           |
| 6                                  | -4                                       | 56           | 14           | -2           |
| 7                                  | -4                                       | 55           | 15           | -2           |
| 8                                  | -4                                       | 54           | 16           | -2           |
| 9                                  | -5                                       | 53           | 18           | -2           |
| 10                                 | -6                                       | 52           | 20           | -2           |

| Posición de muestra fraccionaria p | coeficientes del filtro de interpolación |              |              |              |
|------------------------------------|------------------------------------------|--------------|--------------|--------------|
|                                    | fc[ p ][ 0 ]                             | fc[ p ][ 1 ] | fc[ p ][ 2 ] | fc[ p ][ 3 ] |
| 11                                 | -6                                       | 49           | 24           | -3           |
| 12                                 | -6                                       | 46           | 28           | -4           |
| 13                                 | -5                                       | 44           | 29           | -4           |
| 14                                 | -4                                       | 42           | 30           | -4           |
| 15                                 | -4                                       | 39           | 33           | -4           |
| 16                                 | -4                                       | 36           | 36           | -4           |
| 17                                 | -4                                       | 33           | 39           | -4           |
| 18                                 | -4                                       | 30           | 42           | -4           |
| 19                                 | -4                                       | 29           | 44           | -5           |
| 20                                 | -4                                       | 28           | 46           | -6           |
| 21                                 | -3                                       | 24           | 49           | -6           |
| 22                                 | -2                                       | 20           | 52           | -6           |
| 23                                 | -2                                       | 18           | 53           | -5           |
| 24                                 | -2                                       | 16           | 54           | -4           |
| 25                                 | -2                                       | 15           | 55           | -4           |
| 26                                 | -2                                       | 14           | 56           | -4           |
| 27                                 | -2                                       | 12           | 57           | -3           |
| 28                                 | -2                                       | 10           | 58           | -2           |
| 29                                 | -1                                       | 7            | 60           | -2           |
| 30                                 | 0                                        | 4            | 62           | -2           |
| 31                                 | 0                                        | 2            | 63           | -1           |

El proceso de análisis de Codificación Aritmética Binaria Adaptativa Basada en Contexto (CABAC) para datos de segmento es el siguiente. El proceso de inicialización se invoca cuando se inicia el análisis de la sintaxis de la CTU y la CTU es la primera CTU en un mosaico. Tenga en cuenta que el inicio de los datos de segmento también está cubierto por esta sentencia, ya que cada inicio de los datos de segmento es el inicio de un mosaico.

5

En otro ejemplo, el proceso de escaneo de tramas y mosaicos de CTB es el siguiente. La lista ColWidth[ i ] para i que va desde 0 a num\_tile\_columns\_minus1, inclusive, que especifica el ancho de la i-ésima columna de mosaico en unidades de CTB, se deriva de la siguiente manera:

10

```

if( uniform_tile_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        ColWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) /

```

```

( num_tile_columns_minus1 + 1 ) -
    ( i * PicWidthInCtbsY ) /
(num_tile_columns_minus1 + 1 )
else {
5     ColWidth[ num_tile_columns_minus1 ] = PicWidthInCtbsY (6-1)
    for( i = 0; i < num_tile_columns_minus1; i++ ) {
        ColWidth[ i ] = tile_column_width_minus1[ i ] + 1
        ColWidth[ num_tile_columns_minus1 ] -= ColWidth[ i ]
    }
10 }

```

La lista RowHeight[ j ] para j que va desde 0 a num\_tile\_rows\_minus1, inclusive, que especifica la altura de la j-ésima fila de mosaicos en unidades de CTB, se deriva de la siguiente manera:

```

15 if( uniform_tile_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        RowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) -
            ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
else {
20     RowHeight[ num_tile_rows_minus1 ] = PicHeightInCtbsY (6-2)
    for( j = 0; j < num_tile_rows_minus1; j++ ) {
        RowHeight[ j ] = tile_row_height_minus1[ j ] + 1
        RowHeight[ num_tile_rows_minus1 ] -= RowHeight[ j ]
    }
25 }

```

La lista ColBd[ i ] para i que va desde 0 a num\_tile\_columns\_minus1 + 1, inclusive, que especifica la ubicación del límite de la columna de mosaico i-ésimo en unidades de CTB, se deriva de la siguiente manera:

```

30 for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
    ColBd[ i + 1 ] = ColBd[ i ] + ColWidth[ i ] (6-3)

```

La lista RowBd[ j ] para j que va desde 0 a num\_tile\_rows\_minus1 + 1, inclusive, que especifica la ubicación del límite de fila del mosaico j-ésimo en unidades de CTB, se deriva de la siguiente manera:

```

35 for( RowBd[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
    RowBd[ j + 1 ] = RowBd[ j ] + RowHeight[ j ] (6-4)

```

La lista CtbAddrRsToTs[ctbAddrRs] para ctbAddrRs que va desde 0 hasta PicSizeInCtbsY - 1, inclusive, que especifica la conversión de una dirección CTB en un escaneo de trama CTB de una imagen a una dirección CTB en un escaneo de mosaico, se deriva de la siguiente manera:

```

for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
    tbX = ctbAddrRs % PicWidthInCtbsY
    tbY = ctbAddrRs / PicWidthInCtbsY
45     for( i = 0; i <= num_tile_columns_minus1; i++ )
        if( tbX >= ColBd[ i ] )
            tileX = i
    for( j = 0; j <= num_tile_rows_minus1; j++ ) (6-5)
50         if( tbY >= RowBd[ j ] )
            tileY = j
            CtbAddrRsToTs[ ctbAddrRs ] = 0
            for( i = 0; i < tileX; i++ )
                CtbAddrRsToTs[ ctbAddrRs ] += RowHeight[ tileY ] * ColWidth[ i ]
55         for( j = 0; j < tileY; j++ )
            CtbAddrRsToTs[ ctbAddrRs ] += PicWidthInCtbsY * RowHeight[ j ]
            CtbAddrRsToTs[ ctbAddrRs ] += ( tbY - RowBd[ tileY ] ) * ColWidth[ tileX ] + tbX - ColBd[ tileX ]
        ]
60 }

```

La lista CtbAddrTsToRs[ ctbAddrTs ] para ctbAddrTs que va desde 0 hasta PicSizeInCtbsY - 1, inclusive, que especifica la conversión de una dirección CTB en escaneo de mosaico a una dirección CTB en escaneo de trama CTB de una imagen, se deriva de la siguiente manera:

```

65 for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) (6-6)
    CtbAddrTsToRs[ CtbAddrRsToTs[ ctbAddrRs ] ] = ctbAddrRs

```

La lista `TileId[ ctbAddrTs ]` para `ctbAddrTs` que va desde 0 hasta `PicSizeInCtbsY - 1`, inclusive, que especifica la conversión de una dirección CTB en escaneo de mosaico a un ID de mosaico, se deriva de la siguiente manera:

```

5  for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )
      for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ ) (6-7)
        for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
10      TileId[ CtbAddrRsToTs[ y * PicWidthInCtbsY + x ] ] = tileIdx

```

La lista `NumCtusInTile[ tileIdx ]` para `tileIdx` que va de 0 a `PicSizeInCtbsY - 1`, inclusive, que especifica la conversión de un índice de mosaico al número de CTU en el mosaico, se deriva de la siguiente manera:

```

15  for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ ) (6-8)
      NumCtusInTile[ tileIdx ] = ColWidth[ i ] * RowHeight[ j ]

```

El conjunto `TileIdToldx[ tileId ]` para un conjunto de valores `tileId` `NumTilesInPic` que especifican la conversión de un ID de mosaico a un índice de mosaico y la lista `FirstCtbAddrTs[ tileIdx ]` para `tileIdx` que va desde 0 a `NumTilesInPic - 1`, inclusive, que especifica la conversión de un ID de mosaico a la dirección CTB en el escaneo de mosaico del primer CTB en el mosaico se deriva de la siguiente manera:

```

25  for( ctbAddrTs = 0, tileIdx = 0, tileStartFlag = 1; ctbAddrTs < PicSizeInCtbsY; ctbAddrTs++ ) {
    if( tileStartFlag ) {
      TileIdToldx[ TileId[ ctbAddrTs ] ] = tileIdx
      FirstCtbAddrTs[ tileIdx ] = ctbAddrTs (6-9)
      tileStartFlag = 0
    }
30    tileEndFlag = ctbAddrTs == PicSizeInCtbsY - 1 || TileId[ ctbAddrTs + 1 ] != TileId[ ctbAddrTs ]
    if( tileEndFlag ) {
      tileIdx++
      tileStartFlag = 1
35  }
}

```

Los valores de `ColumnWidthInLumaSamples[ i ]`, que especifican el ancho de la columna de mosaico *i*-ésima en unidades de muestras de luma, se establecen igual `ColWidth[ i ] << CtbLog2SizeY` para *i*, que va desde 0 a `num_tile_columns_minus1`, inclusive. Los valores de `RowHeightInLumaSamples[ j ]`, que especifican la altura de la fila de mosaico *j* en unidades de muestras de luma, se establecen igual a `RowHeight[ j ] << CtbLog2SizeY` para *j*, que va desde 0 hasta `num_tile_rows_minus1`, inclusive.

Sintaxis RBSP del conjunto de parámetros de imagen.

Tabla 7

| <code>pic_parameter_set_rbsp( ) {</code>      | Descriptor         |
|-----------------------------------------------|--------------------|
| <code>pps_pic_parameter_set_id</code>         | <code>ue(v)</code> |
| <code>pps_seq_parameter_set_id</code>         | <code>ue(v)</code> |
| <code>transform_skip_enabled_flag</code>      | <code>u(1)</code>  |
| <code>single_tile_in_pic_flag</code>          | <code>u(1)</code>  |
| <code>if( !single_tile_in_pic_flag ) {</code> |                    |
| <code>num_tile_columns_minus1</code>          | <code>ue(v)</code> |
| <code>num_tile_rows_minus1</code>             | <code>ue(v)</code> |

|                                                   |       |
|---------------------------------------------------|-------|
| }                                                 |       |
| if( !single_tile_in_pic_flag ) {                  |       |
| uniform_tile_spacing_flag                         | u(1)  |
| if( !uniform_tile_spacing_flag ) {                |       |
| for( i = 0; i < num_tile_columns_minus1; i++ )    |       |
| tile_column_width_minus1[ i ]                     | ue(v) |
| for( i = 0; i < num_tile_rows_minus1; i++ )       |       |
| tile_row_height_minus1[ i ]                       | ue(v) |
| }                                                 |       |
| tile_boundary_treated_as_pic_boundary_flag        | u(1)  |
| if( !tile_boundary_treated_as_pic_boundary_flag ) |       |
| loop_filter_across_tiles_enabled_flag             | u(1)  |
| }                                                 |       |
| rbp_trailing_bits( )                              |       |
| }                                                 |       |

La sintaxis de la cabecera de segmento es de siguiente manera:

Tabla 8

5

|                                  |            |
|----------------------------------|------------|
| slice_header( ) {                | Descriptor |
| slice_pic_parameter_set_id       | ue(v)      |
| first_slice_in_pic_flag          | u(1)       |
| single_tile_in_slice_flag        | u(1)       |
| if( !first_slice_in_pic_flag )   |            |
| top_left_tile_id                 | u(v)       |
| if( !single_tile_in_slice_flag ) |            |
| bottom_right_tile_id             | u(v)       |
| slice_type                       | ue(v)      |
| if ( slice_type != I )           |            |
| log2_diff_ctu_max_bt_size        | ue(v)      |
| dep_quant_enabled_flag           | u(1)       |

|                                                                       |       |
|-----------------------------------------------------------------------|-------|
| if( !dep_quant_enabled_flag )                                         |       |
| sign_data_hiding_enabled_flag                                         | u(1)  |
| all_tiles_mcts_flag                                                   | u(1)  |
| if(                                                                   |       |
| !tile_boundary_treated_as_pic_boundary_flag && !all_tiles_mcts_flag ) |       |
| slice_boundary_treated_as_pic_boundary_flag                           |       |
| if( !single_tile_in_slice_flag ) {                                    |       |
| offset_len_minus1                                                     | ue(v) |
| for( i = 0; i < NumTilesInSlice - 1; i++ )                            |       |
| entry_point_offset_minus1[ i ]                                        | u(v)  |
| }                                                                     |       |
| byte_alignment( )                                                     |       |
| }                                                                     |       |

- La semántica de la cabecera de segmento es la siguiente. Cuando esté presente, el valor del elemento de sintaxis de la cabecera de segmento slice\_pic\_parameter\_set\_id deberá ser igual en todas las cabeceras de segmento de una imagen codificada. El slice\_pic\_parameter\_set\_id especifica el valor de pps\_pic\_parameter\_set\_id para el PPS en uso. El valor de slice\_pic\_parameter\_set\_id debe estar en el intervalo de 0 a 63, inclusive. El first\_slice\_in\_pic\_flag se establece igual a uno para especificar que el segmento es el primer segmento de la imagen en orden de decodificación. El first\_slice\_in\_pic\_flag se establece igual a cero para especificar que el segmento no es el primer segmento de la imagen en orden de decodificación. El single\_tile\_in\_slice\_flag se establece igual a uno para especificar que solo hay un mosaico en el segmento. single\_picture\_in\_pic\_flag igual a 0 especifica que hay más de un mosaico en el segmento. El top\_left\_tile\_id especifica el ID de mosaico del mosaico ubicado en la esquina superior izquierda del segmento. La longitud de top\_left\_tile\_id es  $\text{Ceil}(\text{Log2}((\text{num\_tile\_rows\_minus1} + 1) * (\text{num\_tile\_columns\_minus1} + 1)))$  bits. El valor de top\_left\_tile\_id no debe ser igual al valor de top\_left\_tile\_id de ninguna otra unidad NAL de segmento codificado de la misma imagen codificada. Cuando no está presente, se infiere que el valor de top\_left\_tile\_id es igual a cero. Cuando hay más de un segmento en una imagen, el orden de decodificación de los segmentos en la imagen deberá ser en valor creciente de top\_left\_tile\_id. El bottom\_right\_tile\_id especifica el ID de mosaico ubicado en la esquina inferior derecha del segmento. La longitud de bottom\_right\_tile\_id es  $\text{Ceil}(\text{Log2}((\text{num\_tile\_rows\_minus1} + 1) * (\text{num\_tile\_columns\_minus1} + 1)))$  bits. Cuando no está presente, se infiere que el valor de bottom\_right\_tile\_id es igual a top\_left\_tile\_id. Las variables NumTileRowsInSlice, NumTileColumnsInSlice, and NumTilesInSlice se derivan de la siguiente manera:

```

deltaTileIdx = TileIdToldx[ bottom_right_tile_id ] - TileIdToldx[ top_left_tile_id ]
NumTileRowsInSlice = ( deltaTileIdx / ( num_tile_columns_minus1 + 1 ) ) + 1 (7-25)
NumTileColumnsInSlice = ( deltaTileIdx % ( num_tile_columns_minus1 + 1 ) ) + 1
NumTilesInSlice = NumTileRowsInSlice * NumTileColumnsInSlice

```

- El all\_tiles\_mets\_flag se establece igual a uno para especificar que todos los mosaicos en el segmento son parte de un MCTS, que solo contiene los mosaicos en el segmento para la unidad de acceso actual, y los límites del MCTS (que están coubicados con los límites del segmento) se tratan igual que los límites de imagen. El all\_tiles\_mcts\_flag se establece igual a cero para especificar que lo anterior, tal como se especifica por all\_tiles\_mcts\_flag igual a 1, puede o no aplicarse. El slice\_boundary\_treated\_as\_pic\_boundary\_flag se establece igual a uno para especificar que cada límite de segmento se trata igual que el límite de imagen en el proceso de decodificación. El slice\_boundary\_treated\_as\_pic\_boundary\_flag se establece igual a cero para especificar que cada límite de mosaico no se trata igual que el límite de imagen en el proceso de decodificación. Cuando no está presente, se infiere que el valor de slice\_boundary\_treated\_as\_pic\_boundary\_flag es igual a uno. Cada subconjunto incluirá todos los bits codificados de todas las CTU en el segmento que estén dentro del mismo mosaico.

La FIG. 7 es un diagrama esquemático que ilustra una imagen 700 de ejemplo con límites de segmento definidos por la posición del mosaico. Por ejemplo, la imagen 700 puede emplearse para implementar una imagen 600. Por ejemplo, una imagen 700 puede codificarse y decodificarse a partir de un flujo de bits 500, por ejemplo, por un sistema 200 de códec, un codificador 300 y/o un decodificador 400. Además, la imagen 700 se puede particionar para soportar la codificación y decodificación según el método 100.

La imagen 700 incluye una pluralidad de segmentos que incluyen un primer segmento 721, que puede ser sustancialmente similar a un segmento 521 y/o 621. Otros segmentos pueden ser similares al primer segmento 721, pero no se representan para mayor claridad del análisis. El primer segmento 721 incluye una pluralidad de mosaicos 723 designados con ID 724 de mosaico correspondientes, que pueden ser sustancialmente similares a los mosaicos 523 y los ID 524 de mosaico, respectivamente. Los mosaicos 723 comprenden cada uno a una o más CTU 727 designadas por direcciones 729 de CTU correspondientes, que pueden ser sustancialmente similares a las CTU 527 y las direcciones 529 de CTU, respectivamente. La pluralidad de segmentos son cada uno rectangulares, pero pueden no extenderse completamente desde la pared lateral izquierda de la imagen hasta la pared lateral derecha de la imagen. Por ejemplo, la pared lateral izquierda del primer segmento 721 no es congruente con la pared lateral izquierda del fotograma de la imagen 700 y la pared lateral derecha del primer segmento 721 no es congruente con la pared lateral derecha del fotograma de la imagen 700.

Como se muestra, el primer segmento 721 es rectangular e incluye un número entero de mosaicos 723 (por ejemplo, ningún mosaico 723 parcial/fraccionario). Los mosaicos 723 incluyen un mosaico 723a superior izquierdo ubicado en la esquina superior izquierda del primer segmento 721 y un mosaico 723b inferior derecho ubicado en la esquina inferior derecha del primer segmento 721. Como se señaló anteriormente, los ID 724 de mosaico aumentan en orden de trama. El orden de la trama es de izquierda a derecha y de arriba a abajo en la imagen 700. Además, los ID 724 de mosaico son únicos en la imagen 700. En consecuencia, los límites del primer segmento 721 se pueden identificar de forma única por señalización del mosaico 723a superior izquierdo y el mosaico 723b inferior derecho en el flujo de bits. Esto tiene varias ventajas sobre la señalización de los límites del primer segmento 721 en relación con las coordenadas de la imagen 700. Por ejemplo, el decodificador no necesita alterar el direccionamiento relacionado con el primer segmento 721 cuando no se envía toda la imagen 700 al decodificador. Esto puede ocurrir cuando una región de interés que incluye el primer segmento 721 se envía al decodificador para su visualización y la región de interés no está ubicada en la esquina superior izquierda de la imagen 700 (por ejemplo, la posición relativa del primer segmento 721 es diferente en el codificador y el decodificador).

Además, los ID 724 de mosaico de todos los mosaicos 723 en el primer segmento 721 se pueden determinar basándose en los ID 724 de mosaico 723 del mosaico superior izquierdo y del mosaico 723b inferior derecho. Por ejemplo, `top_left_tile_id` y `bottom_right_tile_id` se pueden emplear en la cabecera de segmento para contener los ID 724 de mosaico del mosaico 723 superior izquierdo y del mosaico 723b inferior derecho. El decodificador puede entonces determinar el número de mosaicos 723 basándose en el mosaico 723 superior izquierdo y el mosaico 723b inferior derecho según la ecuación 7-25 anterior. En el caso de que los ID 724 de mosaico se establezcan como índices para los mosaicos 723, los ID 724 de mosaico se pueden determinar basándose en el número determinado de mosaicos en el primer segmento 721. Cuando los ID 724 de mosaico usan algún otro conjunto de valores, se puede señalar un valor `TileIdToIdx` para convertir entre el índice de mosaico y el ID 724 de mosaico para soportar la determinación como se describe en la ecuación 6-9 anterior.

Además, las CTU 727 se pueden direccionar basándose en el ID 724 de mosaico. Como tal, las direcciones 729 de CTU denotadas como `ctbAddrInTs` o `ctbAddrInRs`, se pueden determinar para cada CTU 727 en cada mosaico 723, por ejemplo empleando la primera dirección CTU 729 (denotada como `FirstCtbAddrTs`) y el ID 724 de mosaico correspondiente (denotado como `CurrTileIdx`) como se describe en la ecuación 6-5 y la tabla 3. Como tal, los límites y la posición del primer segmento 721, los ID 724 de mosaico del primer segmento 721 y las direcciones 729 de CTU del primer segmento 721 se pueden determinar en el decodificador basándose en los ID 724 de mosaico del mosaico 723a superior izquierdo y el mosaico 723b inferior derecho. Como tal, los ID 724 de mosaico del primer segmento 721 y/o las direcciones 729 de CTU del primer segmento 721 se pueden omitir del flujo de bits para soportar una mayor eficiencia de compresión y codificación.

La FIG. 8 es un diagrama esquemático de un dispositivo 800 de codificación de vídeo de ejemplo. El dispositivo 800 de codificación de vídeo es adecuado para implementar los ejemplos/realizaciones descritos como se describe en la presente memoria. El dispositivo 800 de codificación de vídeo comprende puertos 820 de entrada, puertos 850 de salida y/o unidades 810 transceptoras (Tx/Rx), que incluyen transmisores y/o receptores para comunicar datos salida y datos de entrada a través de una red. El dispositivo 800 de codificación de vídeo también incluye un procesador 830 que incluye una unidad lógica y/o una unidad de procesamiento central (CPU) para procesar los datos y una memoria 832 para almacenar los datos. El dispositivo 800 de codificación de vídeo también puede comprender componentes óptico a eléctrico (OE), componentes eléctrico a óptico (EO) y/o componentes de comunicación inalámbrica acoplados a los puertos 850 de salida y/o puertos 820 de entrada para la comunicación de datos a través de redes de comunicación

inalámbricas, ópticas o eléctricas. El dispositivo 800 de codificación de vídeo también puede incluir dispositivos 860 de entrada y/o salida (E/S) para comunicar datos hacia y desde un usuario. Los dispositivos 860 de E/S pueden incluir dispositivos de salida, tales como una pantalla para visualizar datos de vídeo, altavoces para emitir datos de audio, etc. Los dispositivos 860 de E/S también pueden incluir dispositivos de entrada, tales como un teclado, ratón, bola de seguimiento, etc. , y/o interfaces correspondientes para interactuar con tales dispositivos de salida.

El procesador 830 se implementa por hardware y software. El procesador 830 puede implementarse como uno o más chips de CPU, núcleos (por ejemplo, como un procesador multinúcleo), una matriz de puertas programable en campo (FPGA), circuitos integrados de aplicación específica (ASIC), y procesadores de señales digitales (DSP). El procesador 830 está en comunicación con los puertos 820 de entrada, Tx/Rx 810, puertos 850 de salida y la memoria 832. El procesador 830 comprende un módulo 814 de codificación. El módulo 814 de codificación implementa las realizaciones descritas anteriormente, tales como los métodos 100, 900 y 1000, que pueden emplear un flujo de bits 500, una imagen 600 y/o una imagen 700. El módulo 814 de codificación también puede implementar cualquier otro método/mecanismo descrito en la presente memoria. Además, el módulo 814 de codificación puede implementar un sistema 200 de códec, un codificador 300 y/o un decodificador 400. Por ejemplo, el módulo 814 de codificación puede particionar una imagen en segmentos, segmentos en mosaicos, mosaicos en CTU, CTU en bloques y codificar los bloques cuando actúa como un codificador. Además, el módulo 814 de codificación puede señalar los límites de los segmentos de imagen basándose en el ID y posición del mosaico, lo que permite inferir varios datos de segmento, mosaico y CTU y, por lo tanto, omitirlos del flujo de bits para aumentar la eficiencia de la codificación. Al actuar como un decodificador, el módulo 814 de codificación puede reconstruir una imagen e inferir la posición del segmento, los ID de mosaico en el segmento y las direcciones de las CTU basándose en los mosaicos superior izquierdo e inferior derecho en un segmento y, por lo tanto, aumentar la eficiencia de la codificación. Como tal, el módulo 814 de codificación hace que el dispositivo 800 de codificación de vídeo proporcione funcionalidad adicional y/o eficiencia de codificación al particionar y codificar datos de vídeo. Como tal, el módulo 814 de codificación mejora la funcionalidad del dispositivo 800 de codificación de vídeo y además aborda problemas que son específicos de las técnicas de codificación de vídeo. Además, el módulo 814 de codificación efectúa una transformación del dispositivo 800 de codificación de vídeo a un estado diferente. Alternativamente, el módulo 814 de codificación puede implementarse como instrucciones almacenadas en la memoria 832 y ejecutadas por el procesador 830 (por ejemplo, como un producto de programa informático almacenado en un soporte no transitorio).

La memoria 832 comprende uno o más tipos de memoria tales como discos, unidades de cinta, unidades de estado sólido, memoria de solo lectura (ROM), memoria de acceso aleatorio (RAM), memoria flash, memoria ternaria direccionable por contenido (TCAM), memoria de acceso aleatorio estática (SRAM), etc. La memoria 832 se puede usar como un dispositivo de almacenamiento de datos de desbordamiento, para almacenar programas cuando tales programas se seleccionan para su ejecución, y para almacenar instrucciones y datos que se leen durante la ejecución del programa.

La FIG. 9 es un diagrama de flujo de un método 900 de ejemplo de codificación de una imagen, tal como la imagen 600 y/o 700, en un flujo de bits, tal como el flujo de bits 500, basándose en límites de segmento definidos por la posición del mosaico. El método 900 puede ser empleado por un codificador, tal como un sistema 200 de códec, un codificador 300 y/o un dispositivo 800 de codificación de vídeo cuando se realiza el método 100.

El método 900 puede comenzar cuando un codificador recibe una secuencia de vídeo que incluye una pluralidad de imágenes y determina codificar esa secuencia de vídeo en un flujo de bits, por ejemplo basándose en la entrada del usuario. La secuencia de vídeo se particiona en imágenes/fotografías/fotogramas para una posterior división antes de la codificación. En la etapa 901, una imagen se particiona en una pluralidad de segmentos. Los segmentos se particionan en una pluralidad de mosaicos. Los mosaicos se particionan en una pluralidad de CTU que contienen partes de la imagen. Las CTU se particionan además en bloques de codificación. Como se señaló anteriormente, la pluralidad de segmentos son cada uno rectangular. La imagen tiene una pared lateral izquierda y una pared ancha derecha. Además, un primer segmento de la pluralidad de segmentos también tiene una pared lateral izquierda y una pared lateral derecha. El primer segmento puede que no se extienda completamente en la imagen. Por ejemplo, la pared lateral izquierda del primer segmento puede no ser congruente con la pared lateral izquierda de la imagen. Como otro ejemplo, la pared lateral derecha del primer segmento puede no ser congruente con la pared lateral derecha de la imagen. Como otro ejemplo, la pared lateral izquierda del primer segmento y la pared lateral derecha del primer segmento son ambas incongruentes con la pared lateral izquierda de la imagen y la pared lateral derecha de la imagen, respectivamente.

En la etapa 903, se asignan los ID de mosaico a los mosaicos. Las direcciones CTU también se asignan a las CTU basándose en los ID de mosaico. Por ejemplo, a cada uno de los múltiples mosaicos se le puede asociar con/asignar un ID de mosaico. Además, cada dirección CTU se puede asignar a cada CTU basándose en el ID de mosaico que contiene/corresponde a la CTU.

Como el primer segmento es un rectángulo, el primer segmento comprende un mosaico superior izquierdo y un mosaico inferior derecho. En la etapa 905, se determinan el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho del primer segmento. En la etapa 907, el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho se codifican en un flujo de bits (por ejemplo, en una cabecera de segmento) para indicar los límites del primer segmento. Este enfoque se puede usar para codificar la imagen en el flujo de bits para soportar la determinación de los ID de mosaico para toda la pluralidad de mosaicos en el primer segmento por inferencia en un decodificador basándose en el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho.

En la etapa 909, la pluralidad de mosaicos se codifica en el flujo de bits para codificar la imagen. Por ejemplo, los bloques de codificación en las CTU se codifican como valores de predicción y valores residuales transformados. En algunos ejemplos, los ID de mosaico se pueden omitir del flujo de bits para cada uno de los mosaicos en el primer segmento, excepto el mosaico superior izquierdo y el mosaico inferior derecho, al codificar la pluralidad de mosaicos. En algunos ejemplos, la codificación de la pluralidad de mosaicos comprende además omitir las direcciones de una o más de las CTU del flujo de bits. En consecuencia, el flujo de bits se codifica para soportar la determinación de las direcciones de todos los mosaicos y las CTU en el primer segmento por inferencia en un decodificador basándose en el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho. Luego, el flujo de bits se transmite hacia el decodificador. El flujo de bits incluye el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho, lo que soporta la reconstrucción de la imagen en un decodificador como parte de una secuencia de vídeo reconstruida basándose, en parte, en los límites del primer segmento.

La FIG. 10 es un diagrama de flujo de un método 1000 de ejemplo de codificación de una imagen, tal como la imagen 600 y/o 700, de un flujo de bits, tal como el flujo de bits 500, basándose en límites de segmento definidos por la posición del mosaico. El método 1000 puede ser empleado por un decodificador, tal como un sistema 200 de códec, un decodificador 400 y/o un dispositivo 800 de codificación de vídeo cuando se realiza el método 100.

El método 1000 puede comenzar cuando un decodificador comienza a recibir un flujo de bits que incluye datos de imagen codificados que representan una secuencia de vídeo, por ejemplo, como resultado del método 900. En la etapa 1001, el flujo de bits se recibe en el decodificador. Específicamente, el flujo de bits incluye datos de imagen codificados en una pluralidad de segmentos. Los segmentos se particionan en una pluralidad de mosaicos. Los mosaicos se particionan en una pluralidad de CTU. Las CTU se particionan en una pluralidad de bloques de codificación. Como se señaló anteriormente, la pluralidad de segmentos son cada uno rectangular. Los datos de la imagen están asociados con un fotograma de la imagen con una pared lateral izquierda y una pared ancha derecha. Además, un primer segmento de la pluralidad de segmentos también tiene una pared lateral izquierda y una pared lateral derecha. El primer segmento puede que no se extienda completamente en el fotograma de la imagen. Por ejemplo, la pared lateral izquierda del primer segmento puede no ser congruente con la pared lateral izquierda del fotograma de la imagen. Como otro ejemplo, la pared lateral derecha del primer segmento puede no ser congruente con la pared lateral derecha del fotograma de la imagen. Como otro ejemplo, la pared lateral izquierda del primer segmento y la pared lateral derecha del primer segmento son ambas incongruentes con la pared lateral izquierda del fotograma de la imagen y la pared lateral derecha de la imagen del fotograma de la imagen, respectivamente.

En la etapa 1003, se determinan el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho asociados con el primer segmento, por ejemplo, a partir de la cabecera de segmento asociado con el primer segmento. Los límites del primer segmento se pueden determinar basándose en el ID de mosaico superior izquierdo y en el ID de mosaico inferior derecho en la etapa 1005.

El primer segmento puede comprender una pluralidad de mosaicos que incluyen un mosaico superior izquierdo y un mosaico inferior derecho. Además, los mosaicos están asociados cada uno con un ID de mosaico. El flujo de bits puede omitir los ID de los mosaicos para cada uno de la pluralidad de mosaicos distintos del mosaico superior izquierdo y del mosaico inferior derecho. En consecuencia, los ID de mosaico para todos de la pluralidad de mosaicos en el primer segmento se determinan en la etapa 1007 basándose en el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho. Además, cada uno de los mosaicos puede contener una o más CTU que contienen datos de imágenes codificadas. Las direcciones de las CTU se pueden asignar basándose en los ID de mosaico correspondientes. En algunos ejemplos, las direcciones de una o más de las CTU se pueden omitir del flujo de bits. En tales casos, las direcciones de todas las CTU en el primer segmento también se pueden determinar basándose en el ID de mosaico superior izquierdo y en el ID de mosaico inferior derecho.

En la etapa 1009, los segmentos, mosaicos, las CTU y bloques de codificación pueden decodificarse para generar una imagen reconstruida basándose en los límites del primer segmento. La imagen reconstruida también se puede reenviar a una pantalla como parte de una secuencia de vídeo reconstruida.

La FIG. 11 es un diagrama esquemático de un sistema 1100 de ejemplo para codificar una secuencia de vídeo de imágenes, tales como la imagen 600 y/o 700, en un flujo de bits, tales como el flujo de bits 500, basándose en límites de segmento definidos por la posición del mosaico. El sistema 1100 puede implementarse por un codificador y un decodificador, tal como un sistema 200 de códec, un codificador 300, un decodificador 400 y/o un dispositivo 800 de codificación de vídeo. Además, el sistema 1100 se puede emplear al implementar el método 100, 900 y/o 1000.

El sistema 1100 incluye un codificador 1102 de vídeo. El codificador 1102 de vídeo comprende un módulo 1101 de particionamiento para particionar una imagen en una pluralidad de segmentos y particionar cada uno de la pluralidad de segmentos en una pluralidad de mosaicos. El codificador 1102 de vídeo comprende además un módulo 1103 de determinación para determinar un ID de mosaico superior izquierdo y un ID de mosaico inferior derecho asociados con un primer segmento. El codificador 1102 de vídeo comprende además un módulo 1105 de codificación que codifica el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho en un flujo de bits para indicar los límites del primer segmento, y codifica la pluralidad de mosaicos en el flujo de bits para codificar la imagen. El codificador 1102 de vídeo comprende además un módulo 1107 de transmisión para transmitir el flujo de bits que incluye el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho para soportar la reconstrucción de la imagen en un decodificador como parte de una secuencia de vídeo reconstruida basándose, en parte, en los límites del primer segmento. El codificador 1102 de vídeo puede configurarse además para realizar cualquiera de las etapas del método 900.

El sistema 1100 también incluye un decodificador 1110 de vídeo. El decodificador 1110 de vídeo comprende un módulo 1111 de recepción para recibir un flujo de bits que incluye datos de imagen codificados en una pluralidad de segmentos. El decodificador 1110 de vídeo comprende además un módulo 1113 de agrupación en mosaicos para determinar un ID de mosaico superior izquierdo y un ID de mosaico inferior derecho asociados con un primer segmento. El decodificador 1110 de vídeo comprende además un módulo 1115 de determinación de límites para determinar los límites del primer segmento basándose en el ID de mosaico superior izquierdo y el ID de mosaico inferior derecho. El decodificador 1110 de vídeo comprende además un módulo 1117 de decodificación para decodificar la pluralidad de segmentos para generar una imagen reconstruida basándose en los límites del primer segmento. El decodificador 1110 de vídeo puede configurarse además para realizar cualquiera de las etapas del método 1000.

Un primer componente se acopla directamente a un segundo componente cuando no hay componentes intermedios, excepto una línea, una traza u otro medio entre el primer componente y el segundo componente. El primer componente se acopla indirectamente a un segundo componente cuando hay componentes intermedios, excepto una línea, una traza u otro medio entre el primer componente y el segundo componente. El término "acoplado" y sus variantes incluyen tanto los acoplados directamente como los acoplados indirectamente. El uso del término "aproximadamente" significa un intervalo que incluye  $\pm 10\%$  del número posterior a menos que se indique lo contrario.

También debe entenderse que las etapas de los métodos ejemplares expuestos en la presente memoria no necesariamente requieren realizarse en el orden descrito, y el orden de las etapas de tales métodos debe entenderse como meramente ejemplar. Asimismo, se pueden incluir etapas adicionales en tales métodos, y ciertas etapas se pueden omitir o combinar, en métodos consistentes con varias realizaciones de la presente descripción.

Si bien se han proporcionado varias realizaciones en la presente descripción, se puede entender que los sistemas y métodos descritos se pueden incorporar en muchas otras formas específicas. Los presentes ejemplos deben considerarse como ilustrativos y no restrictivos, y la intención no es limitarse a los detalles proporcionados en la presente memoria. Por ejemplo, los varios elementos o componentes podrían combinarse o integrarse en otro sistema o ciertas características podrían omitirse o no implementarse.

Además, las técnicas, sistemas, subsistemas, métodos descritos e ilustrados en las varias realizaciones como discretos o separados podrían combinarse o integrarse con otros sistemas, componentes, técnicas, o métodos sin apartarse del alcance de la presente descripción.

## REIVINDICACIONES

1. Un método implementado en un decodificador, comprendiendo el método
- 5 recibir (1001), por un receptor del decodificador, un flujo de bits que incluye datos de imagen codificados en una pluralidad de segmentos;  
  
determinar (1003), por un procesador (830) del decodificador, un índice (723a) de mosaico superior izquierdo y un índice (723b) de mosaico inferior derecho asociados con un primer segmento;  
10 reconstruir (1009), por el procesador (830), el primer segmento (721) para generar una imagen reconstruida basándose en el índice (723a) de mosaico superior izquierdo asociado con un primer segmento, caracterizado por que;  
  
15 en donde cada uno de los mosaicos contiene una o más unidades de árbol de codificación, CTU, que contienen datos de imágenes codificadas, y en donde las direcciones de las CTU se asignan basándose en los índices de mosaico de los mosaicos (723) correspondientes,  
  
en donde la reconstrucción del primer segmento (721) comprende determinar las direcciones de todas las CTU  
20 en el primer segmento (721) basándose en el índice (723a) de mosaico superior izquierdo y el índice (723b) de mosaico inferior derecho.
2. El método de la reivindicación 1, en donde la pluralidad de segmentos son cada uno rectangulares, en donde el primer segmento (721) tiene un límite lateral izquierdo y un límite lateral derecho, en donde los datos de  
25 imagen están asociados con un fotograma de la imagen con un límite lateral izquierdo y un límite lateral derecho, y en donde el límite lateral izquierdo del primer segmento (721) no es congruente con el límite lateral izquierdo del fotograma de la imagen, el límite lateral derecho del primer segmento (721) no es congruente con el límite lateral derecho del fotograma de la imagen, o combinaciones de los mismos.
- 30 3. El método de la reivindicación 1 o 2, en donde el primer segmento (721) comprende una pluralidad de mosaicos que incluyen un mosaico superior izquierdo y un mosaico inferior derecho, en donde la pluralidad de mosaicos están cada uno asociados con un índice de mosaico, en donde el flujo de bits omite índices de mosaico para cada uno de la pluralidad de mosaicos distintos del mosaico superior izquierdo y el mosaico inferior derecho.
- 35 4. El método de cualquiera de las reivindicaciones 1-3, que comprende además determinar los índices de mosaico para toda la pluralidad de mosaicos en el primer segmento (721) basándose en el índice (723a) de mosaico superior izquierdo y el índice (723b) de mosaico inferior derecho.
- 40 5. El método de la reivindicación 1, en donde las direcciones de una o más de las CTU se omiten del flujo de bits.
6. El método de cualquiera de las reivindicaciones 1-5, que comprende además reenviar, por el procesador (830), la imagen reconstruida hacia una pantalla como parte de una secuencia de vídeo reconstruida.
- 45 7. Un método implementado en un codificador, comprendiendo el método  
  
particionar, por un procesador (830) del codificador, una imagen en una pluralidad de segmentos;  
  
50 particionar, por el procesador (830), cada uno de la pluralidad de segmentos en una pluralidad de mosaicos (723);  
  
determinar, por el procesador (830), un elemento de sintaxis para determinar un índice (723a) de mosaico superior izquierdo y un índice (723b) de mosaico inferior derecho asociados con un primer segmento (721);  
55 codificar, por el procesador (830), el elemento de sintaxis en un PPS de un flujo de bits; y  
  
codificar, por el procesador (830), el primer segmento (721) en el flujo de bits para codificar la imagen, caracterizado por que;  
60 particionar, por el procesador (830), cada uno de los mosaicos (723) en una o más unidades de árbol de codificación, CTU, antes de codificar los mosaicos (723), conteniendo las CTU partes de la imagen; y  
  
asignar, por el procesador (830), direcciones a las CTU basándose en los índices de los mosaicos (723)  
65 correspondientes,

en donde el flujo de bits está codificado para soportar la determinación de las direcciones de todas las CTU en el primer segmento (721) por inferencia en un decodificador basándose en el índice (723a) de mosaico superior izquierdo y el índice (723b) de mosaico inferior derecho.

5 8. El método de la reivindicación 7, en donde la pluralidad de segmentos son cada uno rectangulares, en donde el primer segmento (721) tiene un límite lateral izquierdo y un límite lateral derecho, en donde la imagen está asociada con un límite lateral izquierdo y un límite lateral derecho, y en donde el límite lateral izquierdo del primer segmento (721) no es congruente con el límite lateral izquierdo de la imagen, el límite lateral derecho del primer segmento (721) no es congruente con el límite lateral derecho de la imagen, o combinaciones de los mismos.

10 9. El método de la reivindicación 7 u 8, en donde el primer segmento (721) comprende un mosaico superior izquierdo y un mosaico inferior derecho, en donde la pluralidad de mosaicos (723) están cada uno asociados con un índice de mosaico, en donde la codificación de la pluralidad de mosaicos (723) comprende además omitir los índices de mosaico del flujo de bits para cada uno de los mosaicos (723) en el primer segmento (721) distintos del mosaico superior izquierdo y el mosaico inferior derecho.

15 10. El método de cualquiera de las reivindicaciones 7-9, en donde el flujo de bits está codificado para soportar la determinación de los índices de mosaico para toda la pluralidad de mosaicos (723) en el primer segmento (721) por inferencia en un decodificador basándose en el índice (723a) de mosaico superior izquierdo y el índice (723b) de mosaico inferior derecho.

20 11. El método de cualquiera de las reivindicaciones 7-10, en donde la codificación de la pluralidad de mosaicos (723) comprende además omitir las direcciones de una o más de las CTU del flujo de bits.

25 12. El método de cualquiera de las reivindicaciones 7-11, que comprende además transmitir, por un transmisor del decodificador, el flujo de bits que incluye el índice (723a) de mosaico superior izquierdo y el índice (723b) de mosaico inferior derecho para soportar la reconstrucción de la imagen en un decodificador como parte de una secuencia de vídeo reconstruida.

30 13. Un dispositivo 800 de codificación de vídeo que comprende:

un procesador (830), un receptor acoplado al procesador (830) y un transmisor acoplado al procesador (830), estando el procesador (830), el receptor y el transmisor configurados para realizar el método de cualquiera de las reivindicaciones 1-12.

35 14. Un soporte legible por ordenador no transitorio que comprende un producto de programa informático para su uso por un dispositivo de codificación de vídeo, el producto de programa informático que comprende instrucciones ejecutables por ordenador almacenadas en el soporte legible por ordenador no transitorio de manera que cuando se ejecutan por un procesador (830) hacen que el dispositivo de codificación de vídeo realice el método de cualquiera de las reivindicaciones 1-12.

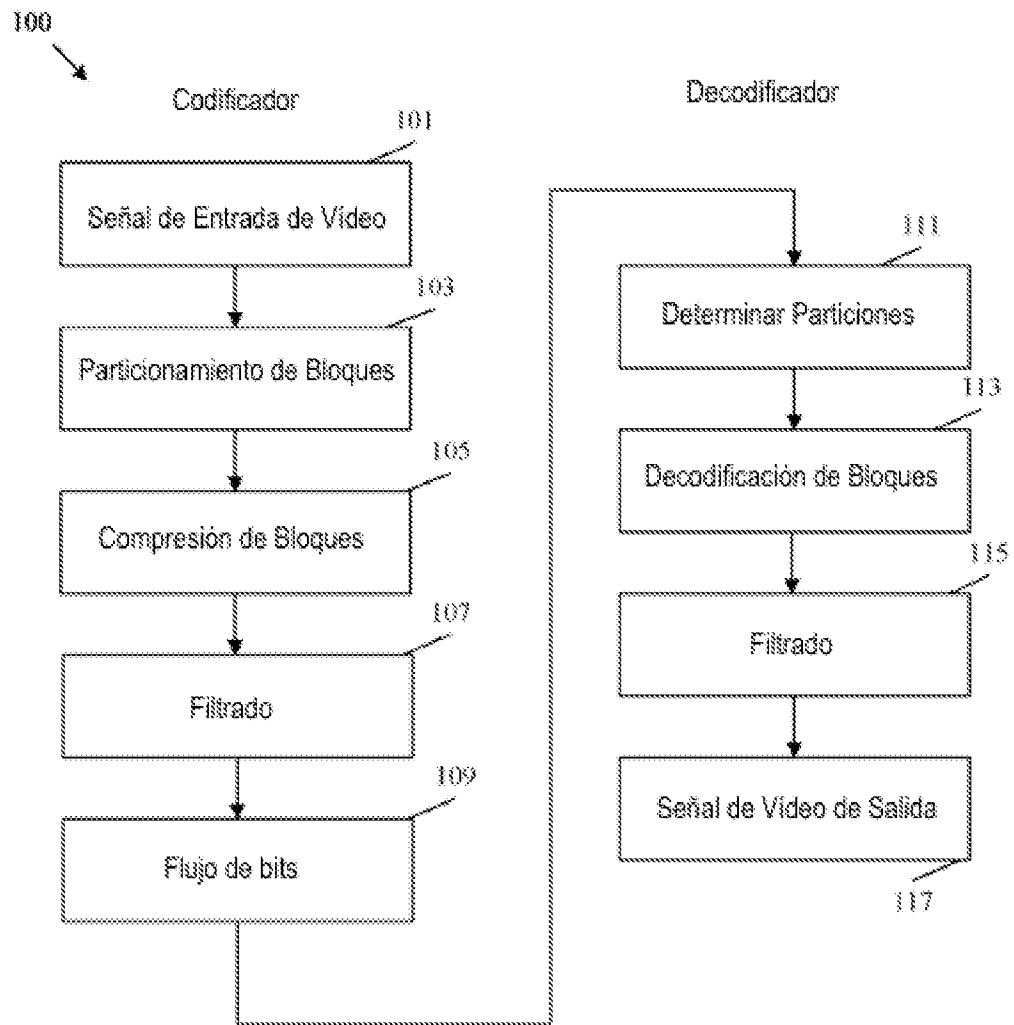


FIG. 1

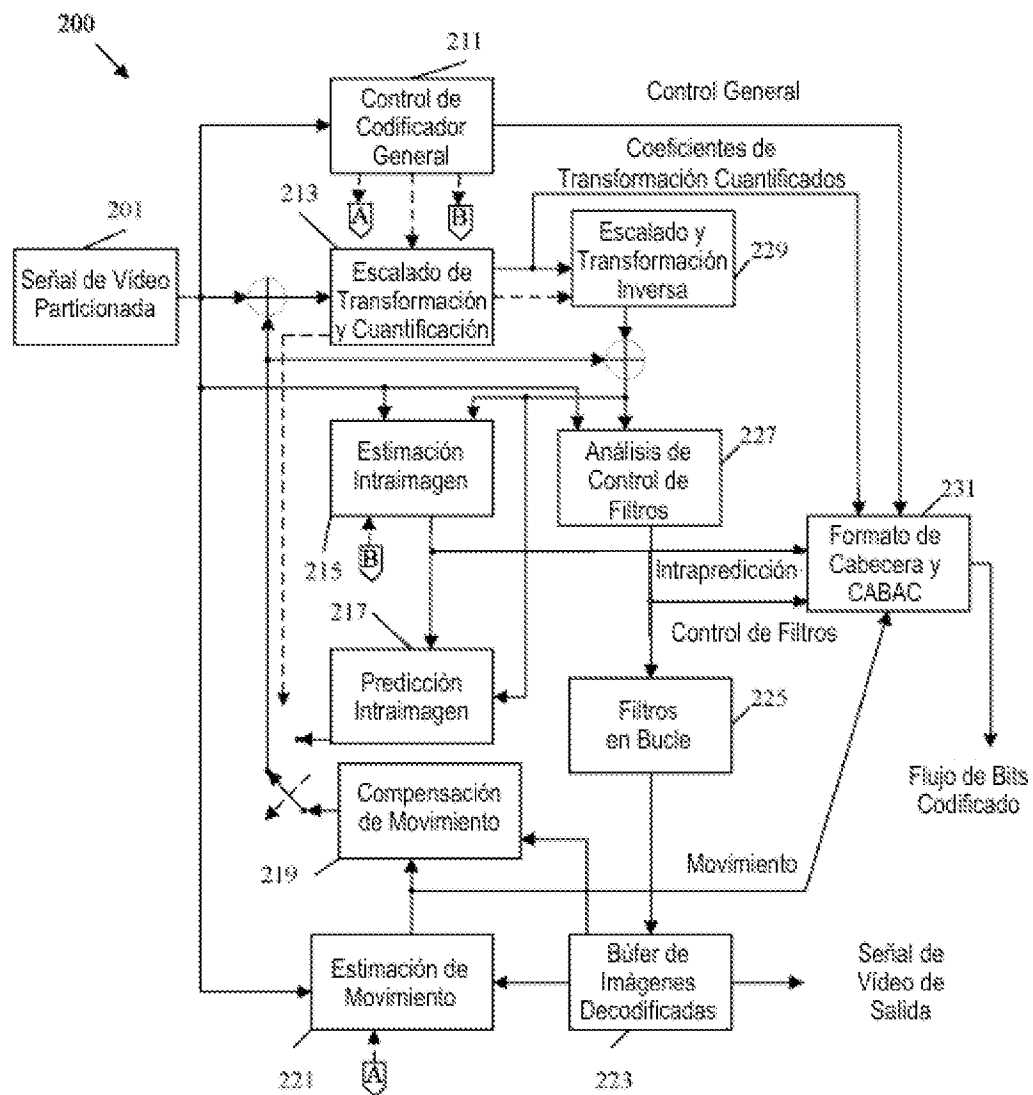


FIG. 2

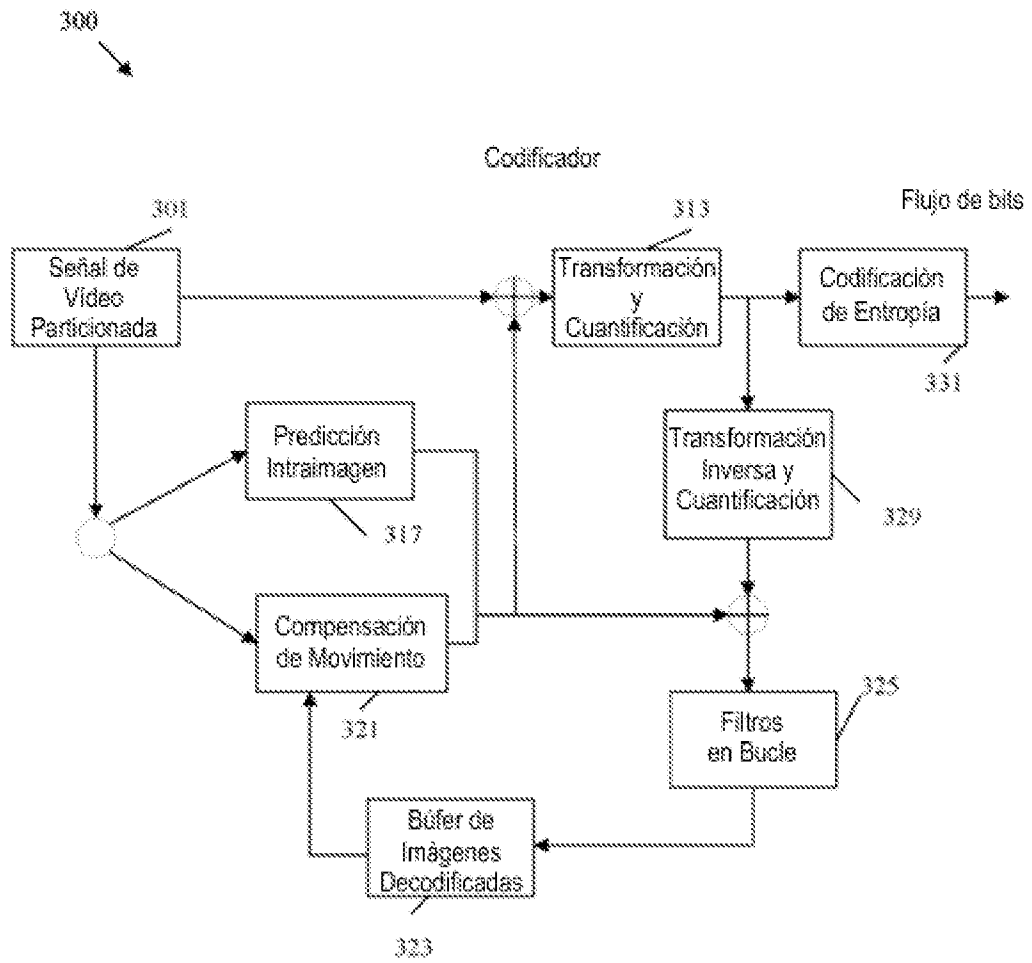


FIG. 3

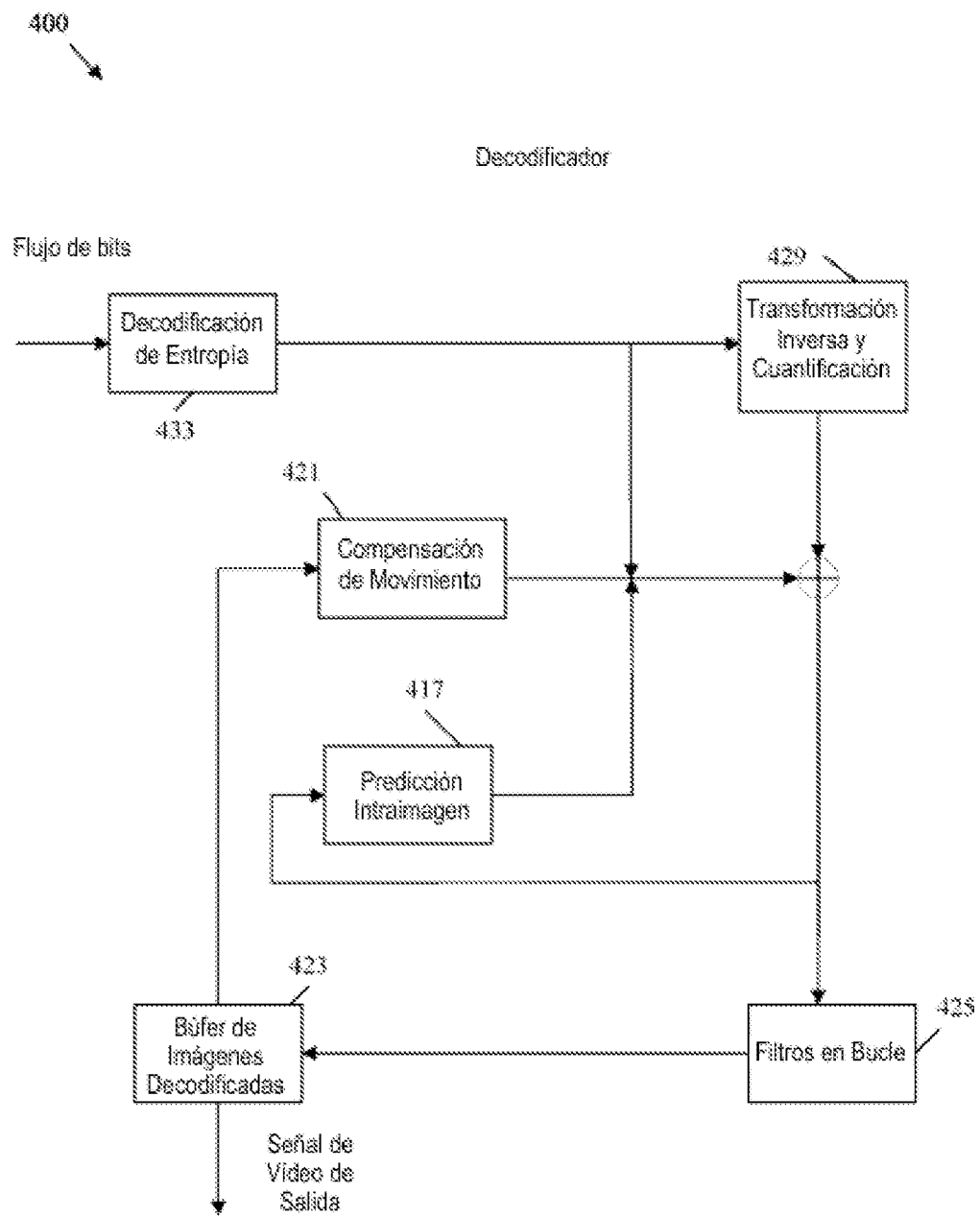


FIG. 4

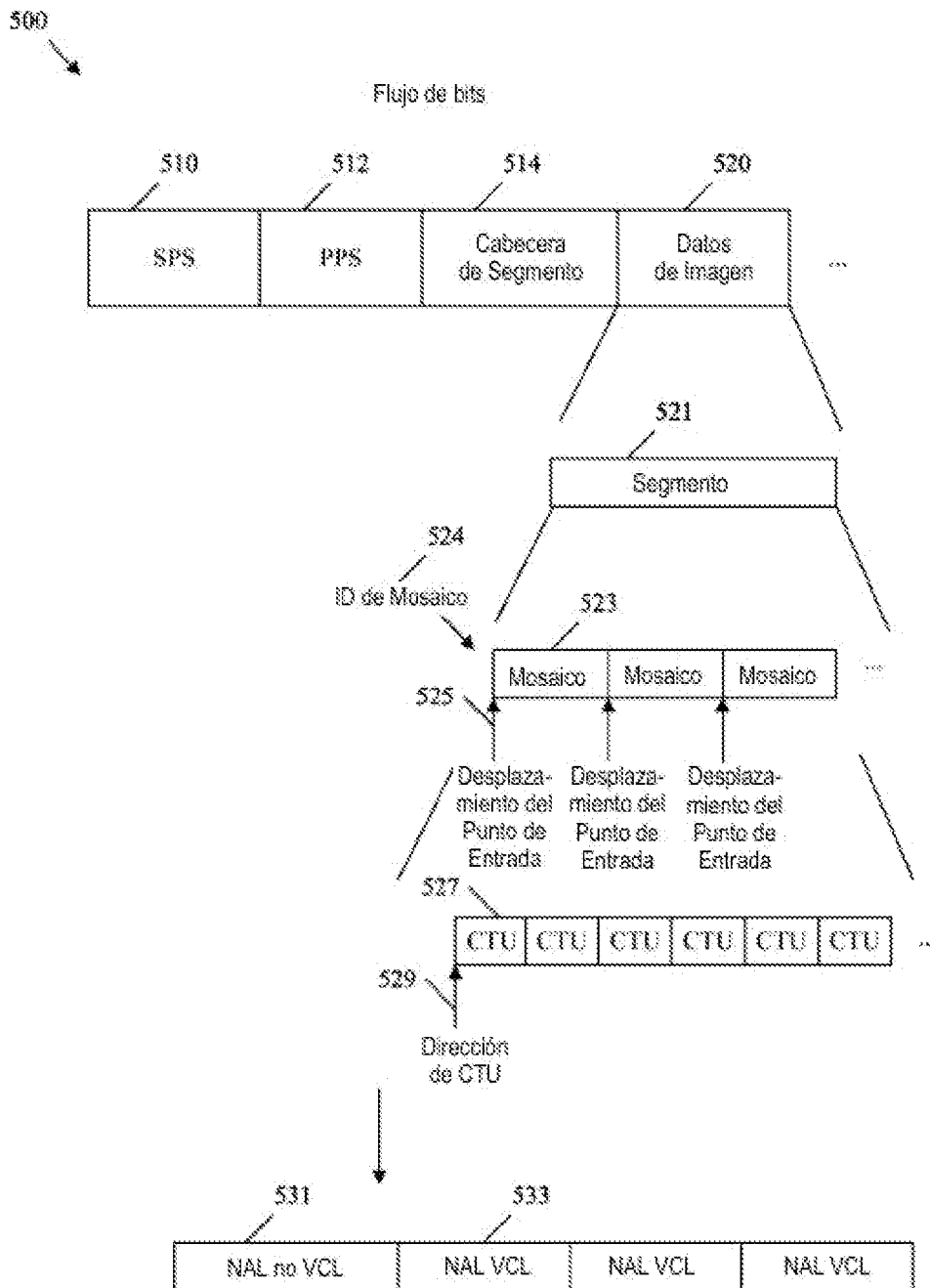


FIG. 5

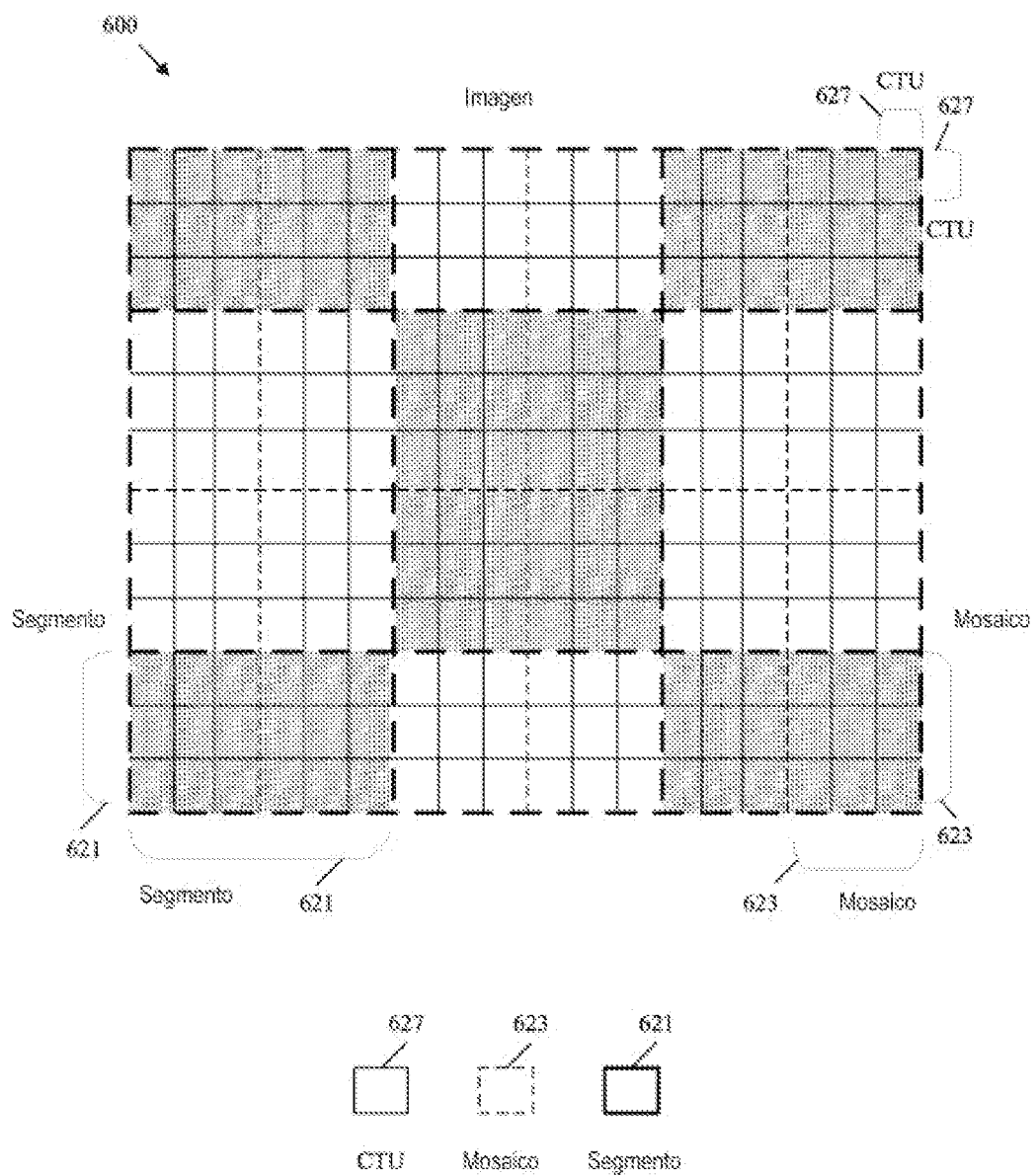


FIG. 6

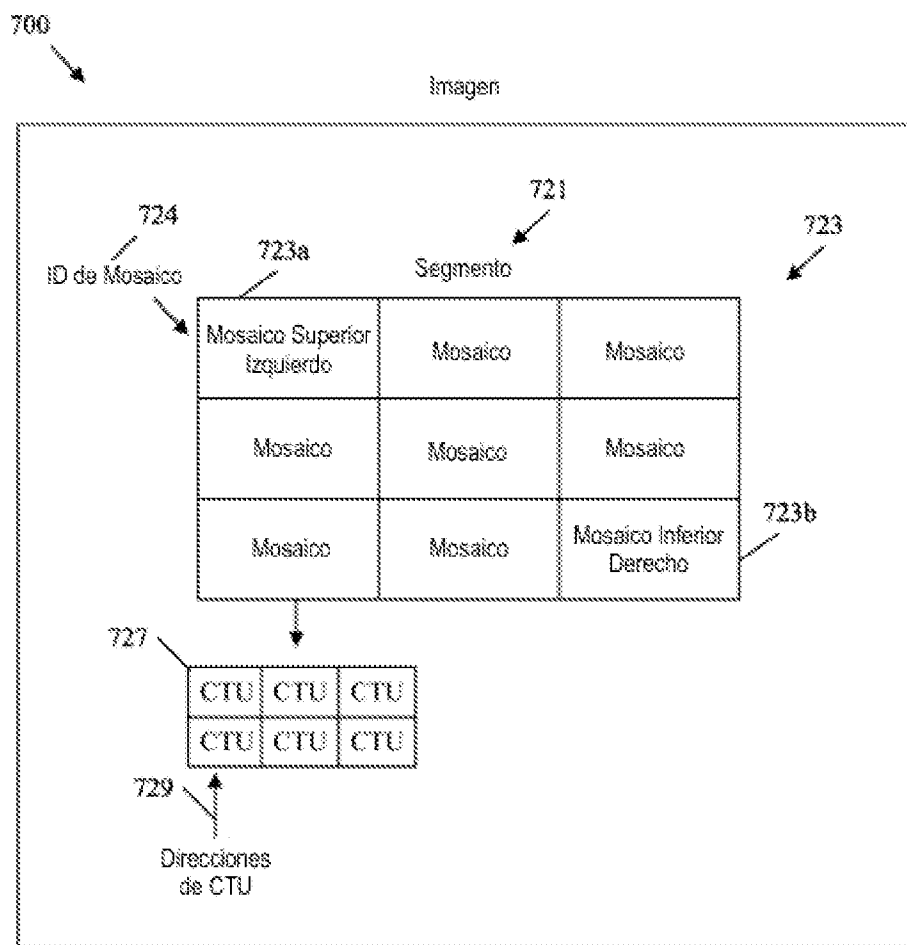


FIG. 7

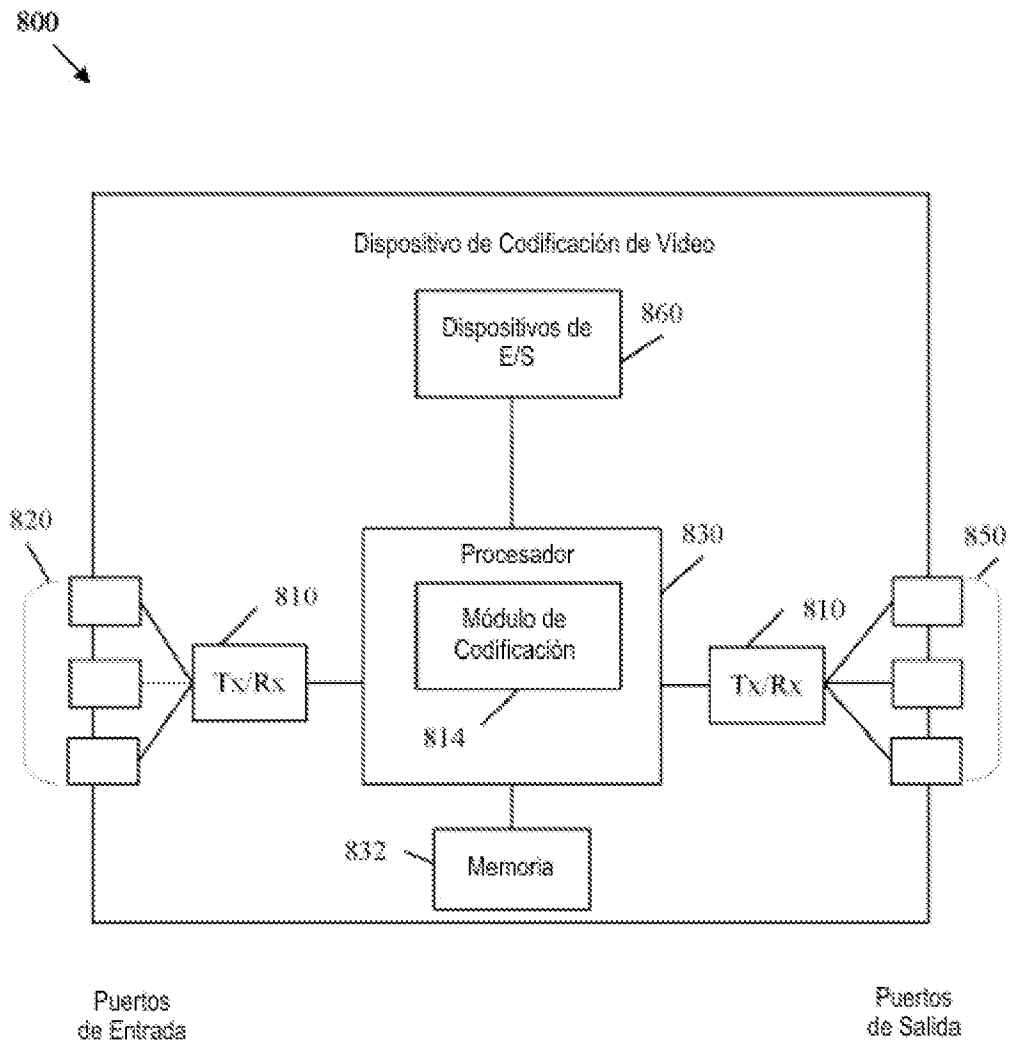


FIG. 8

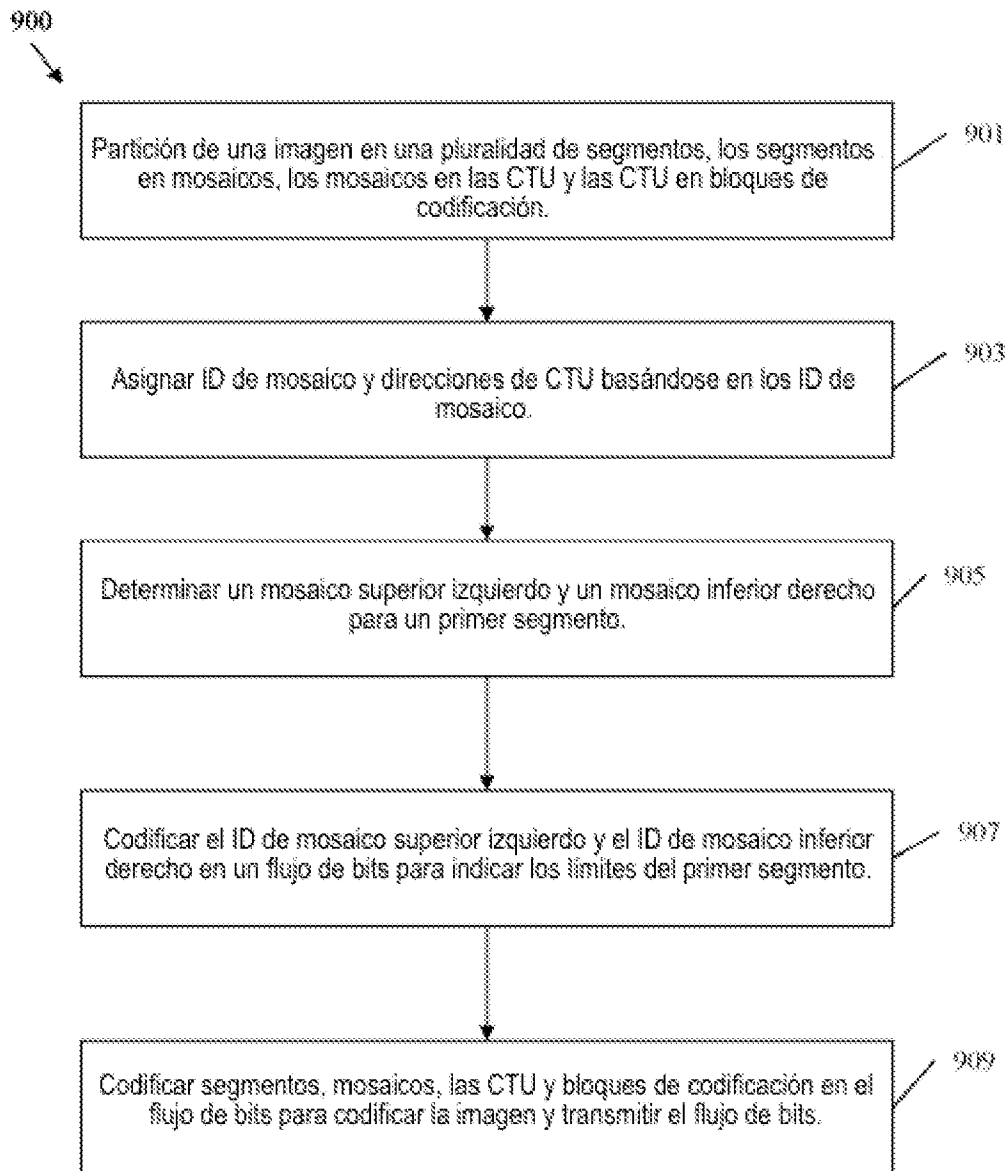


FIG. 9

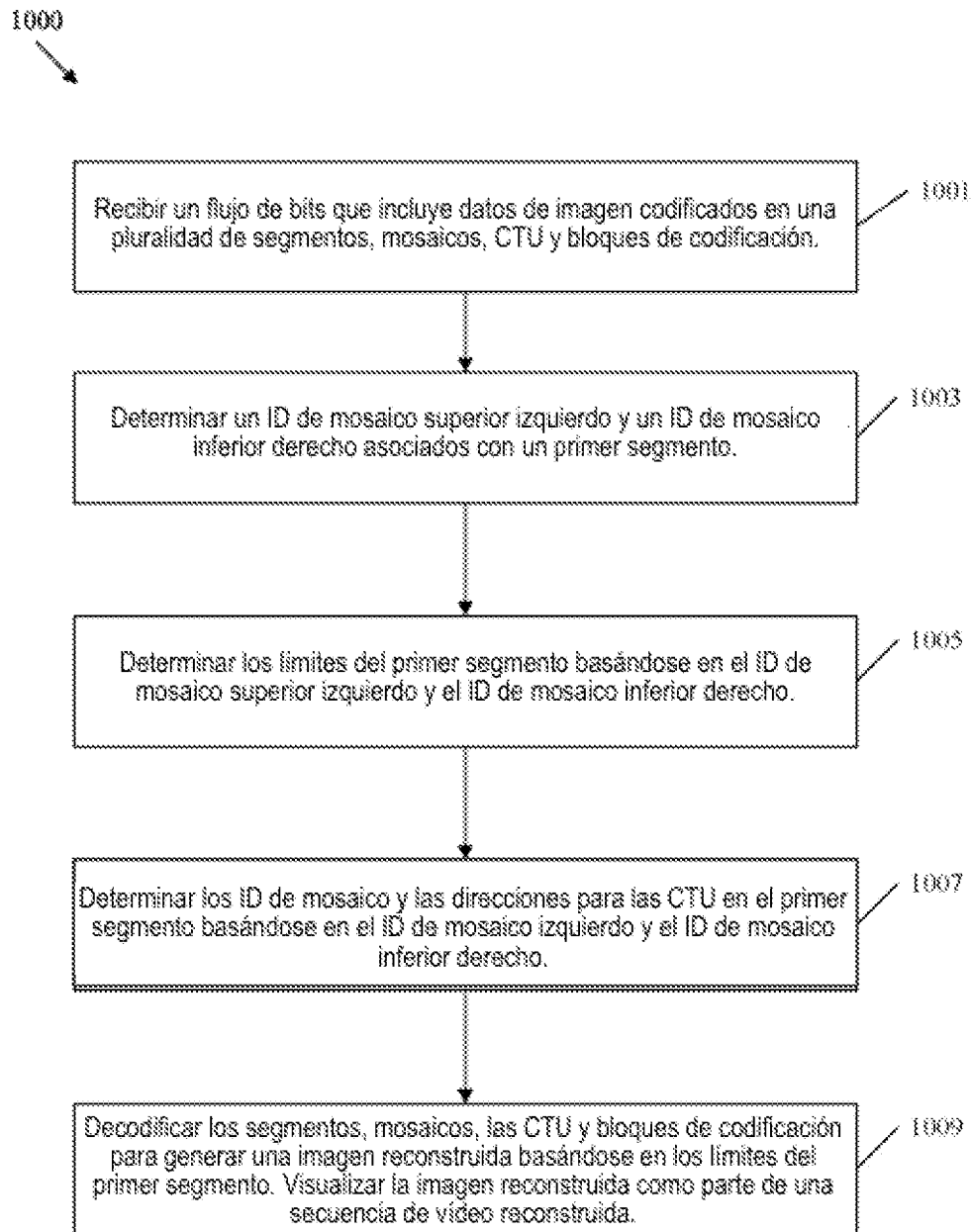


FIG. 10

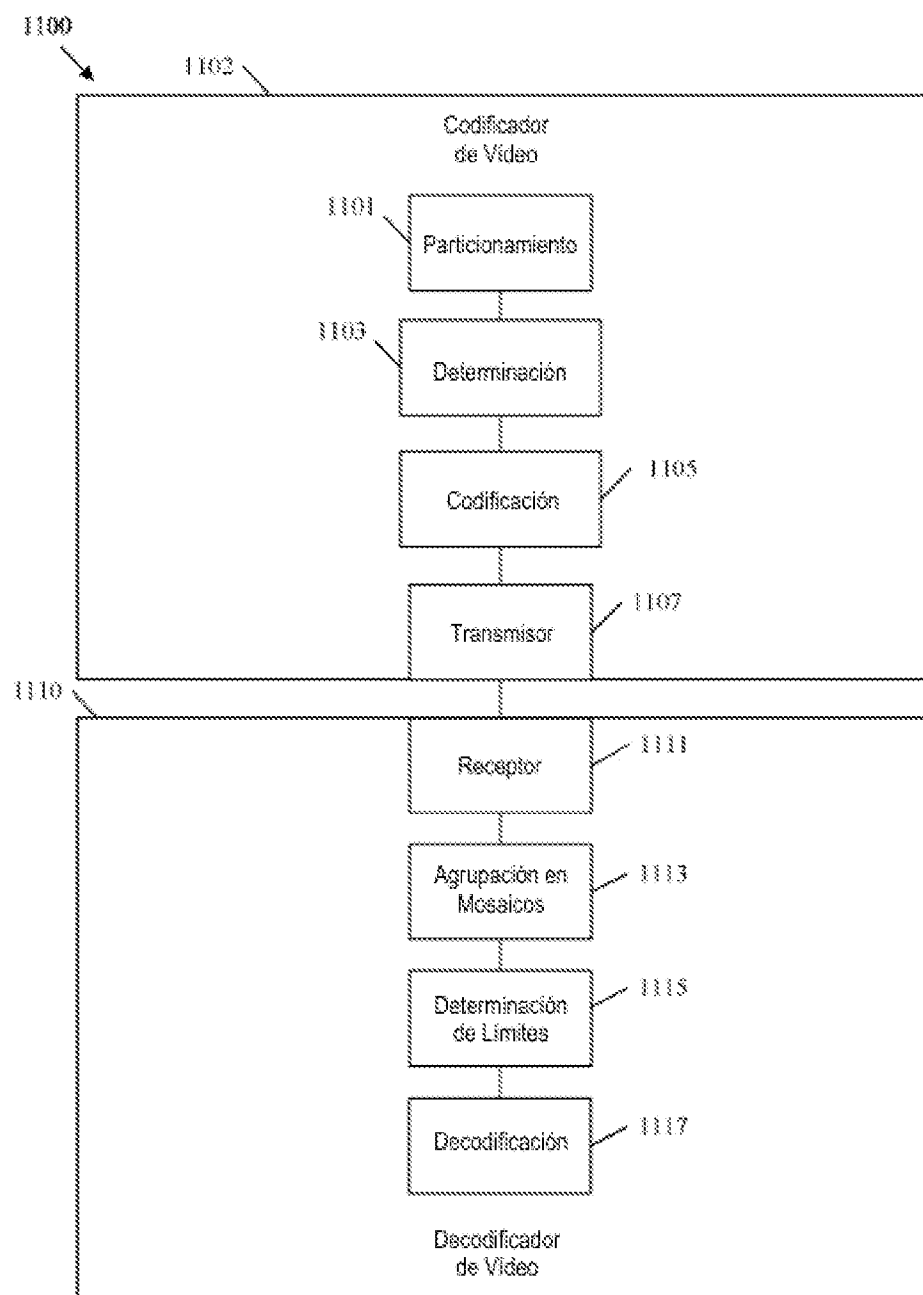


FIG. 11