(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 703 628**

(51) Int. CI.:

H04N 19/105 (2014.01)
H04N 19/70 (2014.01)
H04N 19/51 (2014.01)
H04N 19/513 (2014.01)
H04N 19/30 (2014.01)
H04N 19/463 (2014.01)
H04N 19/187 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **28.02.2013 PCT/KR2013/001659**

(87) Fecha y número de publicación internacional: **06.09.2013 WO13129878**

(96) Fecha de presentación y número de la solicitud europea: **28.02.2013 E 13754230 (4)**

(97) Fecha y número de publicación de la concesión europea: **07.11.2018 EP 2822276**

(54) Título: **Método de predicción intercapa y aparato que hace uso del mismo**

(30) Prioridad:

29.02.2012 US 201261604538 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.03.2019

(73) Titular/es:

LG ELECTRONICS INC. (100.0%)
20, Yeouido-don, Yeongdeungpo-gu
Seoul 150-721, KR

(72) Inventor/es:

KIM, CHULKEUN;
PARK, SEUNGWOOK;
LIM, JAEHYUN;
JEON, YONGJOON;
PARK, JOONYOUNG;
PARK, NAERI y
JEON, BYEONGMOON

(74) Agente/Representante:

MIR PLAJA, Mireia

DESCRIPCIÓN

Método de predicción intercapa y aparato que hace uso del mismo

Campo técnico

La presente invención se refiere a una técnica de compresión de vídeo, y más particularmente, a un método y un aparato para realizar una codificación de vídeo escalable.

Antecedentes de la técnica

En los últimos años, la demanda de vídeos de alta resolución y alta calidad ha aumentado cada vez más en diversos campos de aplicación. Con la mejora de la resolución y la calidad del vídeo, se incrementa también la cantidad de datos del mismo.

Con el incremento de la cantidad de datos, se han desarrollado aparatos que presentan una variedad de rendimientos y redes con diversos entornos.

Con el desarrollo de los aparatos que presentan una variedad de rendimientos y las redes con diversos entornos, el mismo contenido se puede usar con una calidad variada.

Específicamente, puesto que la calidad de vídeo que pueden soportar los terminales se diversifica y se diversifican también los entornos de red construidos, en algunos entornos se pueden usar vídeos con una calidad genérica pero, en otros entornos, se pueden usar vídeos con una calidad mayor.

Por ejemplo, un usuario que haya comprado contenido de vídeo con un terminal móvil puede disfrutar del contenido de vídeo con una pantalla mayor y una resolución más alta usando un módulo de visualización de pantalla grande en su casa.

Con los últimos servicios de difusión con una resolución de alta definición (HD), los usuarios se han acostumbrado a vídeos de alta resolución y alta calidad, y los proveedores de servicios y los usuarios han mostrado interés en servicios de definición ultra-alta (UHD) con una resolución de cuatro o más veces la correspondiente de la HDTV así como en la HDTV.

Por lo tanto, con el fin de proporcionar servicios de vídeo solicitados por usuarios en diversos entornos en función de la calidad, es posible dotar de escalabilidad a la calidad de vídeo, tal como la calidad de imagen de un vídeo, la resolución de un vídeo, el tamaño de un vídeo, y la frecuencia de cuadro de un vídeo, sobre la base de métodos de codificación/descodificación de alta eficiencia para vídeos de alta capacidad.

El documento US 2009/0028245 A1 da a conocer un método para obtener información de codificación para imágenes de alta resolución a partir de imágenes de baja resolución, y dispositivos de codificación y de descodificación que implementan dicho método. De forma más detallada, este documento da a conocer un método de predicción intercapa para obtener información de codificación de por lo menos una parte de imágenes de una alta resolución a partir de información de codificación de por lo menos una parte de imágenes de una baja resolución cuando la relación entre las dimensiones de la parte de imágenes de alta resolución (es decir, anchura y altura) y las dimensiones de la parte de imágenes de baja resolución (es decir, anchura y altura) está vinculada con una relación intercapa comprendida entre 1 y 2. Se pueden usar dos relaciones intercapa diferentes en la dirección horizontal y en la vertical. Cada imagen se divide en macrobloques. La posición de un macrobloque en una imagen se puede identificar o bien mediante uno de sus píxeles, por ejemplo, el píxel superior izquierdo de dicho macrobloque, o bien mediante sus coordenadas en unidades de macrobloques. A un macrobloque de una imagen de baja resolución se le denomina macrobloque de baja resolución. A un macrobloque de una imagen de alta resolución se le denomina macrobloque de alta resolución. Las imágenes de baja resolución se pueden codificar y, por tanto, descodificar de acuerdo con los procesos de codificación/descodificación descritos en la MPEG4 AVC. Cuando se lleva a cabo la codificación, la información de codificación de imágenes de baja resolución se asocia a cada macrobloque en dicha imagen de baja resolución. Esta información de codificación comprende, por ejemplo, información sobre una partición y, posiblemente, una subpartición de los macrobloques, sobre el modo de codificación (por ejemplo, modo de codificación inter, modo de codificación intra ...), posiblemente sobre vectores de movimiento y sobre índices de referencia. Un índice de referencia asociado a un bloque actual de píxeles permite identificar la imagen en la que está ubicado el bloque usado para predecir el bloque actual. De acuerdo con la MPE4-AVC, se usan dos listas de índices de referencia L0 y L1. A continuación, se consideran dos capas espaciales, una capa de base correspondiente a las imágenes de baja resolución y una capa de mejora correspondiente a las imágenes de alta resolución. Las imágenes de baja resolución pueden ser una versión submuestreada de subimágenes de imágenes de la capa de mejora. Las imágenes de baja y alta resolución también pueden ser proporcionadas por cámaras diferentes.

En este caso, las imágenes de baja resolución no se obtienen submuestreando imágenes de alta resolución y se pueden proporcionar parámetros geométricos a través de medios externos (por ejemplo, mediante las propias cámaras). Un macrobloque de baja resolución está asociado a un macrobloque de alta resolución si, cuando se superpone la parte de imagen de baja resolución sobremuestreada por la relación intercapa en ambas direcciones con la parte de imagen de alta resolución, el macrobloque de baja resolución cubre al menos parcialmente dicho macrobloque de la imagen de alta resolución. En el contexto de un proceso de codificación espacialmente escalable, tal como el descrito en JSVM1, se pueden codificar macrobloques de alta resolución usando modos de codificación clásicos (es decir, predicción intra y predicción inter) como aquellos usados para codificar imágenes de baja resolución. Además, algunos macrobloques específicos de las imágenes de alta resolución se pueden usar en un modo de predicción intercapa. Este último modo está solamente autorizado para macrobloques de capa superior cubiertos totalmente por la capa de base escalada. Los macrobloques que no cumplen este requisito únicamente pueden usar los modos clásicos, es decir, modos de predicción intra y predicción inter, mientras que los macrobloques que cumplen este requisito pueden usar los modos o bien de predicción intra, o bien de predicción inter o bien de predicción intercapa.

El documento JCTVC-G683_r2 del *Joint Collaborative Team on Video Coding (*JCT_VC*)* de la ITU-T SG16 WP3 y la ISO/IEC JTC1/SC29/WG11, presenta varios cambios para simplificar el proceso de generación de candidatos fusionables adicionales. En la propuesta, se elimina el candidato de bipredicción escalado, la operación de poda de candidatos fusionables adicionales se limita únicamente a los candidatos de bipredicción combinados, y se añaden candidatos de deriva de movimiento. La simplificación propuesta incluye limitar la operación de poda a candidatos de predicción combinados, lo cual significa que solamente los candidatos de bipredicción combinados requieren operación de poda antes de añadirse a la lista. La totalidad del resto de candidatos (bipredicción escalada, candidato cero y candidato de deriva de mv) no se compararán o podarán incluso la información de movimiento es la misma que algunos candidatos existentes en la lista y la eliminación del candidato de bipredicción escalado. Cuando se encuentra por lo menos un candidato en los vecinos espaciales y temporales, se añade una deriva de mv a cada candidato vecino para generar un nuevo candidato fusionable, con el mismo inter_dir y el mismo refidx. Si los dos cuadros de referencia de L0 y L1 son del mismo lado del cuadro actual (hacia adelante o hacia atrás), el candidato de deriva de mv se genera añadiendo un valor de deriva al vector de movimiento que apunta al cuadro de referencia más alejado del cuadro actual. Cuando no se encuentra ningún candidato en los vecinos espaciales y temporales, se añaden mv=(0, 0) refidx=0 y cuatro de sus versiones de deriva de mv. Cuando el candidato mv=(0, 0) refidx=0 ya se encuentra entre los vecinos espaciales y temporales, se añade el candidato de movimiento cero de mv=(0, 0) refidx=1 antes de añadir candidatos de deriva de movimiento. Con los cambios propuestos que se han descrito anteriormente, los candidatos fusionables adicionales, finales, incluyen: Clase de candidato 1: Candidato de bipredicción combinado (con operación de poda); Clase de candidato 2: Candidato cero refidx1 (con comparación); Clase de candidato 3: Candidatos de deriva de movimiento (sin operación de poda).

Sumario de la invención

Problemas técnicos

Es un objetivo de la invención proporcionar un método y un aparato capaces de mejorar el efecto de la codificación de vídeo escalable y de reducir la cantidad de información que se debe transmitir por duplicado por capas.

Es otro objetivo de la invención proporcionar un método y un aparato capaces de mejorar los efectos de codificación/descodificación para una capa de mejora usando información de una capa de base.

Es todavía otro objetivo de la invención proporcionar un método y un aparato capaces de utilizar información de una capa de base en función de diversas escalabilidades.

Es todavía otro objetivo de la invención proporcionar un método y un aparato capaces de mejorar la eficiencia de codificación usando varios métodos de predicción intercapa.

Es todavía otro objetivo de la invención proporcionar un método y un aparato capaces de mejorar la eficiencia de codificación para una capa de mejora usando por lo menos una de información de textura, información de movimiento, información de sintaxis, información de unidades, información de parámetros, información residual e información diferencial de una capa de base.

Solución a los problemas

Los anteriores objetivos se logran con la combinación de características de las reivindicaciones independientes. Las reivindicaciones dependientes definen realizaciones preferidas.

De acuerdo con un aspecto de la invención, se proporciona un método de predicción intercapa. El método incluye obtener información de movimiento intercapa a partir de una capa de referencia y predecir un bloque actual en una capa actual usando la información de movimiento intercapa. La información de movimiento intercapa incluye un vector de

movimiento intercapa obtenido a partir de la capa de referencia. en este caso, el vector de movimiento intercapa se obtiene escalando un vector de movimiento de la capa de referencia sobre la base de una relación de resolución de la capa de referencia y la capa actual.

De acuerdo con otro aspecto de la invención, se proporciona un descodificador de vídeo escalable. El descodificador de vídeo escalable incluye un primer módulo de predicción que predice una capa de referencia y un segundo módulo de predicción que predice un bloque actual en una capa actual usando información de movimiento intercapa sobre la base de la predicción del primer módulo de predicción. En este caso, la información de movimiento intercapa incluye un vector de movimiento intercapa obtenido a partir de la capa de referencia, y el segundo módulo de predicción escala un vector de movimiento de la capa de referencia sobre la base de una relación de resolución entre la capa de referencia y la capa actual.

Efectos ventajosos

De acuerdo con la invención, es posible mejorar el efecto de la codificación de vídeo escalable y reducir la cantidad de información que se debe transmitir de manera duplicada por capas.

De acuerdo con la invención, es posible mejorar los efectos de codificación/descodificación para una capa de mejora usando información de una capa de base.

De acuerdo con la invención, es posible utilizar información de una capa de base en función de diversas escalabilidades.

De acuerdo con la invención, es posible mejorar la eficiencia de codificación usando diversos métodos de predicción intercapa.

Breve descripción de los dibujos

La FIG. 1 es un diagrama de bloques que ilustra esquemáticamente un codificador de vídeo que soporta escalabilidad de acuerdo con una realización de la invención.

La FIG. 2 es un diagrama de bloques que ilustra un ejemplo de predicción intercapa en el codificador de vídeo que realiza una codificación escalable de acuerdo con la invención.

La FIG. 3 es un diagrama de bloques que ilustra esquemáticamente un descodificador de vídeo que soporta escalabilidad de acuerdo con una realización de la invención.

La FIG. 4 es un diagrama de bloques que ilustra un ejemplo de predicción intercapa en el descodificador de vídeo que realiza una codificación escalable de acuerdo con la invención.

La FIG. 5 es un diagrama que ilustra esquemáticamente un ejemplo de predicción intercapa intra de acuerdo con la invención.

La FIG. 6 es un diagrama que ilustra esquemáticamente un ejemplo de reescalado (submuestreo/sobremuestreo) que se aplica en el transcurso de la predicción intercapa intra de acuerdo con la invención.

La FIG. 7 es un diagrama que ilustra esquemáticamente un ejemplo de sobremuestreo desplazado en fase de acuerdo con la invención.

La FIG. 8 es un diagrama que ilustra un ejemplo de un método que usa un Interpolation_filter_indicator (Indicador_filtro_interpolación) de acuerdo con la invención, es decir, un método de muestreo cuando el valor de Interpolation_filter_indicator es 10.

La FIG. 9 es un diagrama que ilustra brevemente un ejemplo de candidatos de información de movimiento que se usa para realizar una predicción inter en una capa sin referencia a otra capa.

La FIG. 10 es un diagrama de flujo que ilustra un ejemplo de un método de ejecución de una predicción de movimiento intercapa de acuerdo con la invención.

La FIG. 11 es un diagrama que ilustra esquemáticamente un método de obtención de información de movimiento de una capa de referencia de acuerdo con la invención.

La FIG. 12 es un programa que ilustra esquemáticamente un método de escalado de un mvIL de acuerdo con la invención.

La FIG. 13 es un diagrama que ilustra brevemente un ejemplo de un método de ejecución de una predicción de sintaxis intercapa de acuerdo con la invención.

La FIG. 14 es un diagrama que ilustra esquemáticamente un ejemplo de un método de aplicación de predicción residual intercapa de acuerdo con la invención.

La FIG. 15 es un diagrama que ilustra esquemáticamente un ejemplo de predicción de información de unidades intercapa de acuerdo con la invención.

La FIG. 16 es un diagrama que ilustra un ejemplo de aplicación de predicción de unidades intercapa de acuerdo con la invención.

La FIG. 17 es un diagrama que ilustra esquemáticamente otro ejemplo de la predicción de unidades intercapa de acuerdo con la invención.

La FIG. 18 es un diagrama que ilustra esquemáticamente todavía otro ejemplo de la predicción de unidades intercapa de acuerdo con la invención.

La FIG. 19 es un diagrama que ilustra brevemente un ejemplo de un método de ejecución de una predicción de textura intercapa de acuerdo con la invención.

La FIG. 20 es un diagrama que ilustra esquemáticamente un ejemplo de predicción de parámetros de filtros intercapa de acuerdo con la invención.

La FIG. 21 es un diagrama que ilustra esquemáticamente un método de ejecución de una predicción intra cuando se aplica un modo diferencial intercapa de acuerdo con la invención.

Descripción de realizaciones

La presente invención se puede modificar variadamente de diversas formas, y se describirán realizaciones específicas de la misma y estas se mostrarán en los dibujos. No obstante, las realizaciones no están destinadas a limitar la invención. Los términos usados en la siguiente descripción se usan meramente para describir realizaciones específicas, pero no están destinados a limitar la invención. La expresión de un número singular incluye la expresión del número plural, siempre que se interprete claramente de forma diferente. Los términos tales como “incluir” y “tener” están destinados a indicar que existen características, números, etapas, operaciones, elementos, componentes, o combinaciones de los mismos usados en la siguiente descripción, y debe interpretarse, por tanto, que no se excluye la posibilidad de existencia o adición de una o más características, números, etapas, operaciones, elementos, componentes o combinaciones diferentes de los mismos.

Por otro lado, los elementos en los dibujos descritos en la invención se dibujan de manera independiente por motivos de comodidad explicativa de las diferentes funciones específicas en un aparato de codificación/descodificación de imágenes y ello no significa que los elementos se materialicen con hardware independiente o software independiente. Por ejemplo, dos o más elementos de los elementos se pueden combinar para formar un único elemento, o un elemento se puede dividir en diversos elementos. Las realizaciones en las que se combinan y/o dividen los elementos pertenecen al alcance de la invención sin desviarse con respecto al concepto de esta última.

En lo sucesivo en la presente, se describirán de manera detallada realizaciones ejemplificativas de la invención en referencia a los dibujos adjuntos. Los componentes equivalentes en los dibujos se indicarán con números de referencia equivalentes y no se describirán de manera repetida.

En un método de codificación de vídeo que soporte escalabilidad (al que, en lo sucesivo en la presente, se hará referencia como “codificación escalable”), las señales de entrada se pueden procesar por capas. En función de las capas, las señales de entrada (vídeos de entrada) pueden ser diferentes entre sí en por lo menos uno de resolución, frecuencia de cuadro, profundidad de bits, formato de color y relación de aspecto.

En esta descripción, la codificación escalable incluye codificación escalable y descodificación escalable.

En la codificación/descodificación escalable, es posible reducir la transmisión/procesado duplicado de información y mejorar la eficiencia de compresión ejecutando una predicción intercapa que use una diferencia entre capas, es decir, sobre la base de la escalabilidad.

La FIG. 1 es un diagrama de bloques que ilustra esquemáticamente un codificador de vídeo que soporta escalabilidad de acuerdo con una realización de la invención.

En referencia a la FIG. 1, el codificador 100 de vídeo incluye un módulo 105 de codificación para la capa 1 y un módulo 155 de codificación para la capa 0.

La capa 0 puede ser una capa de base, una capa de referencia, o una capa inferior, y la capa 1 puede ser una capa de mejora, una capa actual o una capa superior.

El módulo 105 de codificación para la capa 1 incluye un módulo 110 de predicción inter/intra, un módulo 115 de transformada/cuantificación, un módulo 120 de filtrado, una memoria intermedia de imágenes descodificadas (DPB) 125, un módulo 130 de codificación entrópica, un módulo 135 de predicción de parámetros de unidades, un módulo 140 de predicción/reescalado de movimiento, un módulo 145 de predicción/reescalado de textura, un módulo 150 de predicción de parámetros, y un multiplexor (MUX) 185.

El módulo 155 de codificación para la capa 0 incluye un módulo 160 de predicción inter/intra, un módulo 165 de transformada/cuantificación, un módulo 170 de filtrado, una DPB 175, y un módulo 180 de codificación entrópica.

Los módulos 110 y 160 de predicción inter/intra pueden llevar a cabo una predicción inter y una predicción intra sobre un vídeo de entrada, respectivamente. Los módulos 110 y 160 de predicción inter/intra pueden llevar a cabo la predicción por unidades de proceso predeterminadas. La unidad de proceso para la predicción puede ser una unidad de codificación (CU), una unidad de predicción (PU), o puede ser una unidad de transformada (TU).

Por ejemplo, los módulos 110 y 160 de predicción inter/intra pueden determinar cuál de entre la predicción inter o predicción intra aplicar en las unidades de CU, pueden determinar el modo de predicción en las unidades de PU, y pueden llevar a cabo una predicción en las unidades de PU o TU. La predicción a llevar a cabo incluye la construcción de un bloque predicho y la construcción de un bloque residual (señal residual).

En la predicción inter, la predicción se puede llevar a cabo sobre la base de información de por lo menos una de entre una imagen previa y/o una imagen subsiguiente de una imagen actual para construir un bloque predicho. En la predicción intra, la predicción se puede llevar a cabo sobre la base de información de píxeles en una imagen actual para construir un bloque predicho.

Ejemplos del modo o método de predicción inter incluyen un modo de omisión, un modo de fusión, un método de predicción por vectores de movimiento (MVP). En la predicción inter, se puede seleccionar una imagen de referencia para una PU actual de la cual se va a realizar la predicción, y, a partir de la imagen de referencia, se puede seleccionar un bloque de referencia correspondiente a la PU actual. El módulo 160 de predicción inter/intra puede construir un bloque predicho sobre la base del bloque de referencia.

El bloque predicho se puede construir en la unidad de muestras de píxeles enteros o en la unidad de muestras de píxeles inferiores a un píxel entero. En este caso, el vector de movimiento también se puede expresar en la unidad de muestras de píxeles enteros o en la unidad de muestras de píxeles inferiores a un píxel entero.

La información de movimiento en la predicción inter, es decir, información tal como un índice de imagen de referencia, un vector de movimiento, y una señal residual, se codifica por entropía y se transmite a un descodificador de vídeo. Cuando se aplica el modo de omisión, la señal residual no se puede generar, ni transformar, ni cuantificar ni transmitir en absoluto.

Los modos de predicción en la predicción intra pueden incluir 33 modos de predicción direccionales y por lo menos dos modos no direccionales. Los modos no direccionales pueden incluir un modo de predicción DC y un modo plano. En la predicción intra, se puede construir un bloque predicho después de que se aplique un filtro a una muestra de referencia.

Una PU puede ser un bloque con diversos tamaños y formas. Por ejemplo, en el caso de la predicción inter, una PU puede ser bloques con tamaños tales como 2Nx2N, 2NxN, Nx2N, y NxN (donde N es un entero). En el caso de la predicción intra, una PU puede ser bloques con tamaños tales como 2Nx2N y NxN (donde N es un entero). Se puede establecer que una PU con un tamaño de NxN se aplique solamente a un caso específico. Por ejemplo, se puede establecer que la PU con un tamaño de NxN se use solamente para la CU más pequeña o se puede establecer que la misma se use solamente para la predicción intra. Además de las PUs con los tamaños antes mencionados, se pueden definir y usar adicionalmente PUs tales como un bloque de NxmN, un bloque de mNxN, un bloque de 2NxmN, y un bloque de mNx2N (donde m<1).

Los módulos 115 y 165 de transformada/cuantificación realizan un proceso de transformada sobre el bloque residual en las unidades de TU para generar coeficientes de transformada y cuantifican los coeficientes de transformada.

Un bloque de transformada es un bloque rectangular de muestras y es un bloque en el cual se aplica la misma transformada. El bloque de transformada puede ser una TU y puede tener una estructura de árbol cuaternario.

Los módulos 115 y 165 de transformada/cuantificación pueden realizar el proceso de transformada en función del modo de predicción aplicado al bloque residual y del tamaño del bloque de transformada para generar una matriz bidimensional de coeficientes de transformada. Por ejemplo, cuando la predicción intra se aplica a un bloque residual y el bloque residual tiene una matriz de 4x4, el bloque residual se puede transformar usando una transformada de seno discreta (DST). Si no, el bloque residual se puede transformar usando una transformada de coseno discreta (DCT).

Los módulos 115 y 165 de transformada/cuantificación pueden usar de manera fija una transformada específica con independencia del modo de predicción y del tamaño del bloque de transformada. Por ejemplo, los módulos 115 y 165 de transformada/cuantificación pueden aplicar solamente la DST a todos los bloques de transformada. Los módulos 115 y 165 de transformada/cuantificación pueden aplicar solamente la DCT a todos los bloques de transformada.

Los módulos 115 y 165 de transformada/cuantificación pueden cuantificar los coeficientes de transformada para generar los coeficientes de transformada cuantificados.

Los módulos 115 y 165 de transformada/cuantificación pueden transmitir los coeficientes de transformada cuantificados a los módulos 130 y 180 de codificación entrópica. En este momento, los módulos 115 y 165 de transformada/cuantificación pueden reordenar la matriz bidimensional de los coeficientes de transformada cuantificados obteniendo una matriz unidimensional en un orden de exploración predeterminado, y pueden transmitir la matriz unidimensional reordenada a los módulos 130 y 180 de codificación entrópica. Los módulos 115 y 165 de transformada/cuantificación pueden transmitir el bloque reconstruido generado sobre la base del bloque residual y el bloque predicho a los módulos 120 y 170 de filtrado para la predicción inter sin la aplicación de transformada/cuantificación.

Por otro lado, los módulos 115 y 165 de transformada/cuantificación pueden omitir la transformada y realizar únicamente la cuantificación, o pueden omitir tanto la transformada como la cuantificación, si así fuera necesario. Por ejemplo, los módulos 115 y 165 de transformada/cuantificación pueden omitir la transformada para un bloque al que se le haya aplicado un método de predicción específico o que tenga un tamaño específico, o para un bloque al que se le haya aplicado un bloque de predicción específico y tenga un tamaño específico.

Los módulos 130 y 180 de codificación entrópica pueden ejecutar una codificación entrópica sobre los coeficientes de transformada codificados. Para la codificación entrópica se puede usar un método de codificación, tal como un método de Golomb exponencial y una codificación aritmética binaria adaptativa según el contexto (CABAC).

Los módulos 120 y 170 de filtrado pueden aplicar un filtro antibloques, un filtro de bucle adaptativo (ALF) o una compensación adaptativa por muestras (SAO) a una imagen reconstruida.

El filtro antibloques elimina una distorsión de bloque generada en el límite entre bloques en la imagen reconstruida. La ALF lleva a cabo un proceso de filtrado sobre la base de los valores resultado de la comparación de la imagen original con la imagen deconstruida cuyos bloques se filtran con el filtro antibloques. La SAO reconstruye diferencias de compensación entre los bloques residuales a los que se ha aplicado el filtro antibloques y la imagen original, y la misma se aplica en forma de una compensación por bandas, una compensación por bordes o similares.

Los módulos 120 y 170 de filtrado pueden no aplicar la totalidad del filtro antibloques, el ALF y la SAO, sino que pueden aplicar solamente el filtro antibloques o pueden aplicar solamente el filtro antibloques y el ALF o pueden aplicar solamente el filtro antibloques y la SAO.

Las DPBs 125 y 175 pueden recibir y almacenar el bloque reconstruido o la imagen reconstruida de las unidades 125 y 170 de filtrado. La DPB 125 y 175 puede suministrar el bloque o imagen reconstruido a los módulos 110 y 160 de predicción inter/intra que llevan a cabo la predicción inter.

El MUX 185 puede multiplexar información obtenida a la salida del módulo 180 de codificación entrópica para la capa 0 e información obtenida a la salida del módulo 130 de codificación entrópica para la capa 1, y a las mismas se les puede dar salida en forma de un flujo continuo de bits.

Por otro lado, el módulo 105 de codificación para la capa 1 puede incluir un módulo 135 de predicción de parámetros de unidades, un módulo 140 de predicción/reescalado de movimiento, un módulo 145 de predicción/reescalado de textura y un módulo 150 de predicción de parámetros para predecir por predicción intercapa un vídeo de la capa 1 usando la información de la capa 0.

El módulo 135 de predicción de parámetros inter puede obtener información de unidades (CU, PU, y/o TU) de una capa de base para su uso como información de unidades de una capa de mejora, o puede determinar la información de unidades de la capa de mejora sobre la base de la información de unidades de la capa de base.

El módulo 140 de predicción de movimiento lleva a cabo una predicción de movimiento intercapa. La predicción de movimiento intercapa se denomina también predicción intercapa inter. El módulo 140 de predicción de movimiento puede predecir un bloque actual de una capa actual (capa de mejora) usando la información de movimiento de una capa de referencia (capa de base).

El módulo 140 de predicción de movimiento puede escalar la información de movimiento de la capa de referencia, si fuera necesario.

El módulo 145 de predicción de textura puede realizar una predicción de textura intercapa sobre la base de la información de la capa 0. A la predicción de textura intercapa se le hace referencia también como predicción intercapa intra o predicción de BL (Capa de Base) intra. La predicción de textura se puede usar cuando se reconstruye un bloque de referencia de una capa de referencia. En la predicción de textura intercapa, como valor predicho de un bloque actual en una capa de mejora se puede usar una textura de un bloque de referencia en una capa de referencia. En este caso, la textura del bloque de referencia se puede escalar por sobremuestreo.

El módulo 150 de predicción de parámetros puede obtener un parámetro usado en la capa de base para su reutilización en la capa de mejora, o puede predecir un parámetro para la capa de mejora sobre la base del parámetro usado en la capa de base.

Por otro lado, por comodidad explicativa se describe que el módulo 105 de codificación para la capa 1 incluye el MUX 185, aunque el MUX puede ser un dispositivo o módulo independiente del módulo 105 de codificación para la capa 1 y del módulo 155 de codificación para la capa 0.

La FIG. 2 es un diagrama de bloques que ilustra un ejemplo de predicción intercapa en el codificador de vídeo que lleva a cabo una codificación escalable de acuerdo con la invención.

En referencia a la FIG. 2, un módulo 210 de predicción para la capa 1 incluye un módulo 220 de predicción inter/intra y un módulo 230 de predicción intercapa.

El módulo 210 de predicción para la capa 1 puede llevar a cabo una predicción intercapa necesaria para predecir la capa 1 a partir de la información de la capa 0.

Por ejemplo, el módulo 230 de predicción intercapa puede recibir la información de la capa 0 desde el módulo 250 de predicción inter/intra y/o del módulo 260 de filtrado para la capa 0, y puede realizar la predicción intercapa necesaria para predecir la capa 1.

El módulo 220 de predicción inter/intra para la capa 1 puede realizar una predicción inter o una predicción intra usando la información de la capa 1.

El módulo 220 de predicción inter/intra para la capa 1 puede realizar una predicción basándose en la información de la capa 0 con el uso de la información transmitida desde el módulo 230 de predicción intercapa.

Adicionalmente, el módulo 240 de filtrado para la capa 1 puede realizar una operación de filtrado sobre la base de la información de la capa 0, o puede realizar una operación de filtrado sobre la base de la información de la capa 1. La información de la capa 0 se puede transmitir desde el módulo 260 de filtrado para la capa 0 hacia el módulo 240 de filtrado para la capa 1, o se puede transmitir desde el módulo 230 de predicción intercapa para la capa 1 al módulo 240 de filtrado para la capa 1.

Por otro lado, la información transmitida desde la capa 0 al módulo 230 de predicción intercapa puede ser por lo menos una de información sobre parámetros de unidades de la capa 0, información de movimiento de la capa 0, información de textura de la capa 0 e información de parámetros de filtros de la capa 0.

Por consiguiente, el módulo 230 de predicción intercapa puede incluir una parte o la totalidad del módulo 135 de predicción de parámetros de unidades, el módulo 140 de predicción de movimiento, el módulo 145 de predicción de textura y el módulo 150 de predicción de parámetros que llevan a cabo la predicción intercapa en la FIG. 1.

En la capa 1, el módulo 220 de predicción inter/intra se puede corresponder con el módulo 110 de predicción inter/intra de la FIG. 1, y el módulo 240 de filtrado se puede corresponder con el módulo 120 de filtrado de la FIG. 1. En la capa 0, el módulo 250 de predicción inter/intra se puede corresponder con el módulo 160 de predicción inter/intra de la FIG. 1, y el módulo 260 de filtrado se puede corresponder con el módulo 170 de filtrado de la FIG. 1.

La FIG. 3 es un diagrama de bloques que ilustra esquemáticamente un descodificador de vídeo que soporta escalabilidad de acuerdo con una realización de la invención.

En referencia a la FIG. 3, el descodificador 300 de vídeo incluye un módulo 310 de descodificación para la capa 1 y un módulo 350 de descodificación para la capa 0.

La capa 0 puede ser una capa de base, una capa de referencia, o una capa inferior, y la capa 1 puede ser una capa de mejora, una capa actual o una capa superior.

El módulo 310 de descodificación para la capa 1 puede incluir un módulo 315 de descodificación entrópica, un módulo 320 de reordenación, un módulo 325 de descuantificación, un módulo 330 de transformada inversa, un módulo 335 de predicción, un módulo 340 de filtrado y una memoria 345.

El módulo 350 de descodificación para la capa 0 puede incluir un módulo 355 de descodificación entrópica, un módulo 360 de reordenación, un módulo 365 de descuantificación, un módulo 370 de transformada inversa, un módulo 375 de predicción, un módulo 380 de filtrado, y una memoria 385.

Cuando se transmite desde el codificador de vídeo un flujo continuo de bits que incluye información de vídeo, un DEMUX 305 puede desmultiplexar la información por capas y puede transmitir la información a descodificadores por capas.

Los módulos 315 y 355 de descodificación entrópica pueden llevar a cabo una descodificación entrópica de manera que se corresponda con el método de codificación entrópica usado en el codificador de vídeo. Por ejemplo, cuando en el codificador de vídeo se usa la CABAC, los módulos 315 y 355 de descodificación entrópica pueden realizar la descodificación entrópica usando la CABAC.

La información para construir un bloque predicho a partir de la información descodificada por los módulos 315 y 355 de descodificación entrópica se puede suministrar a los módulos 335 y 375 de predicción, y los valores residuales, es decir, los coeficientes de transformada cuantificados, sometidos a la descodificación entrópica por los módulos 315 y 355 de descodificación entrópica se pueden introducir en los módulos 320 y 360 de reordenación.

Los módulos 320 y 360 de reordenación pueden reordenar la información del flujo continuo de bits, es decir, los coeficientes de transformada cuantificados, sometidos a la descodificación entrópica por los módulos 315 y 355 de descodificación entrópica, sobre la base del método de reordenación usado por el codificador de vídeo.

Por ejemplo, los módulos 320 y 360 de reordenación pueden reordenar los coeficientes de transformada cuantificados en una matriz unidimensional para obtener los coeficientes en una matriz bidimensional nuevamente. Los módulos 320 y 360 de reordenación pueden llevar a cabo una operación de exploración sobre la base del modo de predicción aplicado al bloque actual (bloque de transformada) y/o del tamaño del bloque de transformada, para construir una matriz bidimensional de coeficientes (coeficientes de transformada cuantificados).

Los módulos 325 y 365 de descuantificación pueden llevar a cabo una operación de descuantificación sobre la base del parámetro de cuantificación transmitido desde el codificador de vídeo y de los valores de coeficientes reordenados del bloque para generar coeficientes de transformada.

Los módulos 325 y 365 de descuantificación pueden no descuantificar los valores residuales descodificados entrópicamente, sino que pueden transmitir los valores residuales a los módulos 330 y 370 de transformada inversa en función de una condición predeterminada o en función del método de cuantificación usado por el codificador de vídeo.

Los módulos 330 y 370 de transformada inversa pueden llevar a cabo una transformada inversa de la transformada realizada por el módulo de transformada del codificador de vídeo sobre los coeficientes de transformada. Los módulos 330 y 370 de transformada inversa pueden llevar a cabo una DCT inversa y/o una DST inversa de la DCT y DST realizadas por el codificador de vídeo.

La DCT y/o la DST en el codificador de vídeo se pueden realizar selectivamente en función de múltiples informaciones, tales como el método de predicción, el tamaño del bloque actual, y la dirección de la predicción, y los módulos 330 y 370 de transformada inversa del descodificador de vídeo pueden llevar a cabo la transformada inversa sobre la base de la información de transformada usada por el codificador de vídeo.

Por ejemplo, los módulos 330 y 370 de transformada inversa pueden realizar la DCT inversa y la DST inversa en función del modo de predicción/el tamaño del bloque. Por ejemplo, los módulos 330 y 370 de transformada inversa pueden realizar la DST inversa sobre un bloque de luma de 4x4 en el cual se haya aplicado la predicción intra.

Los módulos 330 y 370 de transformada inversa pueden usar de manera fija un método de transformada inversa específico con independencia del modo de predicción/del tamaño del bloque. Por ejemplo, los módulos 330 y 370 de transformada inversa pueden llevar a cabo solamente la DST inversa sobre todos los bloques de transformada. Los

módulos 330 y 370 de transformada inversa pueden llevar a cabo solamente la DCT inversa sobre todos los bloques de transformada.

Los módulos 330 y 370 de transformada inversa pueden transformar inversamente los coeficientes de transformada o un bloque de los coeficientes de transformada para construir una señal residual o un bloque residual.

Los módulos 330 y 370 de transformada inversa pueden omitir la transformada si fuera necesario o en función del método de codificación usado por el codificador de vídeo. Por ejemplo, los módulos 330 y 370 de transformada inversa pueden omitir la transformada para un bloque al que se haya aplicado un método de predicción específico o que tenga un tamaño específico, o a un bloque que tenga un método de predicción específico y que tenga un tamaño específico.

Los módulos 335 y 375 de predicción pueden construir un bloque predicho del bloque actual sobre la base de información de construcción de bloques predichos suministrada desde los módulos 315 y 355 de descodificación entrópica y del bloque descodificado previamente y/o información de imagen suministrada desde las memorias 345 y 385.

Cuando el modo de predicción del bloque actual es un modo de predicción intra, los módulos 335 y 375 de predicción pueden llevar a cabo la predicción intra sobre el bloque actual basándose en información de píxeles de la imagen actual.

Cuando el modo de predicción para el bloque actual es un modo de predicción inter, los módulos 335 y 375 de predicción pueden llevar a cabo la predicción inter sobre el bloque actual sobre la base de información incluida en por lo menos una de una imagen previa y una imagen subsiguiente de la imagen actual. Una parte o la totalidad de la información de movimiento necesaria para la predicción inter se puede obtener en función de la información recibida desde el codificador de vídeo.

Cuando el modo de omisión se usa como modo de predicción inter, el valor residual no se puede transmitir desde el codificador de vídeo y el bloque predicho se puede usar como bloque reconstruido.

Por otro lado, el módulo 335 de predicción para la capa 1 puede realizar la predicción inter o la predicción intra usando solamente la información de la capa 1, y puede realizar la predicción intercapa usando información de otra capa (capa 0).

Por ejemplo, el módulo 335 de predicción para la capa 1 puede predecir el bloque actual usando una de entre información de movimiento de la capa 1, información de textura de la capa 1, información de unidades de la capa 1 e información de parámetros de la capa 1. El módulo 335 de predicción para la capa 1 puede predecir el bloque actual usando múltiples informaciones de entre la información de movimiento de la capa 1, la información de textura de la capa 1, la información de unidades de la capa 1 y la información de parámetros de la capa 1.

El módulo 335 de predicción para la capa 1 puede recibir la información de movimiento de la capa 1 desde el módulo 375 de predicción para la capa 0, y puede realizar la predicción de movimiento. A la predicción de movimiento intercapa se le hace referencia también como predicción intercapa inter. Con la predicción de movimiento intercapa, se puede predecir el bloque actual de la capa actual (capa de mejora) usando la información de movimiento de la capa de referencia (capa de base). El módulo 335 de predicción puede escalar y usar la información de movimiento de la capa de referencia si fuera necesario.

El módulo 335 de predicción para la capa 1 puede recibir la información de textura de la capa 1 desde el módulo 375 de predicción para la capa 0, y puede realizar una predicción de textura intercapa. A la predicción de textura intercapa se le hace referencia también como predicción intercapa intra o predicción de capa base (BL) intra. La predicción de textura intercapa se puede usar cuando se reconstruye un bloque de referencia de una capa de referencia. En la predicción de textura intercapa, como valores predichos de un bloque actual en una capa de mejora se puede usar la textura de un bloque de referencia en una capa de referencia. En este caso, la textura del bloque de referencia se puede escalar por sobremuestreo.

El módulo 335 de predicción para la capa 1 puede recibir la información de parámetros de unidades de la capa 1 desde el módulo 375 de predicción para la capa 0, y puede realizar una predicción de parámetros de unidades. Con la predicción de parámetros de unidades, la información de unidades (CU, PU, y/o TU) de una capa base se puede usar como información de unidades de la capa de mejora, o la información de unidades de la capa de mejora se puede determinar sobre la base de la información de unidades de la capa de base.

El módulo 335 de predicción para la capa 1 puede recibir la información de parámetros de filtrado de la capa 1 desde el módulo 375 de predicción para la capa 0, y puede llevar a cabo una predicción de parámetros. Con la predicción de parámetros, se puede obtener el parámetro usado para la capa de base y el mismo se puede reutilizar para la capa de mejora, o el parámetro de la capa de mejora se puede predecir sobre la base del parámetro usado para la capa de base.

Los sumadores 390 y 395 pueden construir un bloque reconstruido usando el bloque predicho construido por los módulos 335 y 375 de predicción, y el bloque residual construido por los módulos 330 y 370 de transformada inversa. En este caso, los sumadores 390 y 395 se pueden considerar como módulos particulares (módulo de construcción de bloques reconstruidos) que construyen un bloque reconstruido.

El bloque y/o la imagen reconstruida por los sumadores 390 y 395 se pueden suministrar a los módulos 340 y 380 de filtrado.

Los módulos 340 y 380 de filtrado pueden aplicar el filtro antibloques, la SAO y/o el ALF al bloque y/o la imagen reconstruidos.

Los módulos 340 y 380 de filtrado pueden no aplicar la totalidad del filtro antibloques, el ALF y la SAO, sino que pueden aplicar solamente el filtro antibloques o pueden aplicar solamente el filtro antibloques y el LAF o pueden aplicar solamente el filtro antibloques y la SAO.

En referencia al ejemplo ilustrado en la FIG. 3, el módulo 340 de filtrado para la capa 1 puede llevar a cabo una operación de filtrado sobre la imagen reconstruida usando la información de parámetros transmitida desde el módulo 335 de predicción para la capa 1 y/o del módulo 380 de filtrado para la capa 1. Por ejemplo, el módulo 340 de filtrado para la capa 1 puede realizar una operación de filtrado sobre la capa 1 ó una operación de filtrado intercapa usando los parámetros predichos a partir de los parámetros de filtrado aplicados a la capa 0.

Las memorias 345 y 385 pueden almacenar el bloque o imagen reconstruido para su uso como imagen de referencia o bloque de referencia. Las memorias 345 y 385 pueden dar salida a la imagen reconstruida almacenadas en las memorias 345 y 385 por medio de un módulo de salida predeterminado (no ilustrado) o un módulo de visualización (no ilustrado).

En el ejemplo ilustrado en la FIG. 3, el módulo de reordenación, el módulo de descuantificación, el módulo de transformada inversa, y similares se han descrito de manera que son módulos independientes, pero el descodificador de vídeo se puede construir de para conseguir que un módulo del módulo de descuantificación/transformada inversa realice secuencialmente la reordenación, la descuantificación, y la transformada inversa como el codificador de vídeo ilustrado en la FIG. 1.

Por otro lado, el módulo de predicción se ha descrito en referencia a la FIG. 3, pero el módulo de predicción para la capa 1 puede incluir un módulo de predicción intercapa que lleve a cabo un proceso de predicción usando información de otra capa (capa 0) y un módulo de predicción inter/intra que lleve a cabo un proceso de predicción sin usar información de otra capa (capa 0) como se ilustra en la FIG. 1.

La FIG. 4 es un diagrama de bloques que ilustra un ejemplo de predicción intercapa en el descodificador de vídeo que realiza una codificación escalable de acuerdo con la invención.

En referencia a la FIG. 4, un módulo 410 de predicción para la capa 1 incluye un módulo 420 de predicción inter/intra y un módulo 430 de predicción intercapa.

El módulo 410 de predicción para la capa 1 puede realizar una predicción intercapa necesaria para predecir la capa 1 a partir de la información de la capa 0.

Por ejemplo, el módulo 430 de predicción intercapa puede recibir la información de la capa 0 desde el módulo 450 de predicción inter/intra y/o del módulo 460 de filtrado para la capa 0 y puede realizar la predicción intercapa necesaria para predecir la capa 1.

El módulo 420 de predicción inter/intra para la capa 1 puede realizar la predicción inter o la predicción intra usando la información de la capa 1.

El módulo 420 de predicción inter/intra para la capa 1 puede llevar a cabo una predicción basándose en la información de la capa 0 con el uso de la información transmitida desde el módulo 430 de predicción intercapa.

El módulo 440 de filtrado para la capa 1 puede llevar a cabo un filtrado sobre la base de la información de la capa 0 ó puede llevar a cabo un filtrado sobre la base de la información de la capa 1. La información de la capa 0 se puede transmitir desde el módulo 460 de filtrado para la capa 0 hacia el módulo 440 de filtrado para la capa 1, o se puede transmitir desde el módulo 430 de predicción intercapa para la capa 1 hacia el módulo 440 de filtrado para la capa 1.

Por otro lado, la información transmitida desde la capa 0 al módulo 430 de predicción intercapa puede ser al menos una de información sobre parámetros de unidades de la capa 0, información de movimiento de la capa 0, información de textura de la capa 0, e información de parámetros de filtros de la capa 0.

En la capa 1, el módulo 410 de predicción se puede corresponder con el módulo 355 de predicción de la FIG. 3, y el módulo 440 de filtrado se puede corresponder con el módulo 340 de filtrado de la FIG. 3. En la capa 0, el módulo 450 de predicción se puede corresponder con el módulo 375 de predicción de la FIG. 3, y el módulo 460 de filtrado se puede corresponder con el módulo 380 de filtrado de la FIG. 3.

Aunque no se ilustra, el módulo 430 de predicción intercapa puede incluir un módulo de predicción de movimiento, un módulo de predicción de textura, un módulo de predicción de parámetros de unidades, un módulo de predicción de parámetros en función de los tipos de la predicción intercapa que se vaya a llevar a cabo (por ejemplo, predicción de movimiento, predicción de textura, predicción de parámetros de unidades y predicción de parámetros).

En la codificación de vídeo escalable, se puede realizar una predicción intercapa de información de predicción de una capa actual usando información de otra capa. Tal como se describe en referencia a las FIGS. 1 a 4, como ejemplos de la predicción intercapa pueden considerarse la predicción de movimiento, la predicción de textura, la predicción de unidades y la predicción de parámetros.

Los tipos respectivos de predicción intercapa se describirán específicamente a continuación en referencia a los dibujos adjuntos.

Predicción intercapa intra

A la predicción intercapa intra se le hace referencia también como predicción de textura intercapa o predicción de BL (Capa de Base) intra. En esta descripción, por motivos de comodidad en la explicación, los términos de predicción intercapa intra, predicción de textura y predicción de BL intra se pueden intercambiar.

En este momento, para adaptar la imagen reconstruida de una capa de base y una imagen de una capa de mejora entre sí en cuanto a tamaño o resolución de la imagen, la imagen reconstruida de la capa de base se puede someter a sobremuestreo.

Por ejemplo, el sobremuestreo se puede realizar usando un DCTIF (Filtro de Interpolación Basado en DCT). Por ejemplo, muestras de luma se pueden someter al sobremuestreo usando un DCTIF de 8 tomas, y muestras de croma se pueden someter al sobremuestreo usando un DCTIF de 4 tomas.

El sobremuestreo se puede realizar usando interpolación.

Por otro lado, la predicción intercapa intra de acuerdo con la invención presenta características diferentes a las de la técnica relacionada en función de las unidades de codificación/descodificación. Por ejemplo, en el nivel de los bloques de codificación (por ejemplo, CU), la predicción intercapa intra se puede aplicar en función de una partición de CU independiente sin tener en cuenta el tamaño de bloque de una capa de base.

En el nivel de los bloques de predicción (por ejemplo, PU), el coste de la predicción intercapa intra se puede comparar con el coste de la predicción intra en la capa actual (capa de mejora). En este caso, la predicción intercapa intra se compara en cuanto a optimización de distorsión-velocidad (RDO) con la predicción intra usando la imagen reconstruida de la capa de base (capa de referencia) como imagen de referencia sin utilizar el modo de predicción intra.

Para aplicar la predicción intercapa intra, el hecho de la aplicación de la predicción intercapa intra se puede señalizar en forma de una bandera en un nivel superior sin crear un modo nuevo para la PU. Por ejemplo, la información que indica si usar la predicción intercapa intra se puede transmitir en forma de una bandera en una posición subsiguiente a una bandera de división (split_flag) antes de llevar a cabo el análisis sintáctico en un modo de bloque o partición.

En el nivel de bloques de transformada (por ejemplo, TU), la predicción intercapa intra se puede aplicar para llevar a cabo la transformada con la estructura de árbol cuaternario de la HEVC mantenida.

La FIG. 5 es un diagrama que ilustra esquemáticamente un ejemplo en el que se lleva a cabo la predicción intercapa intra de acuerdo con la invención.

En el ejemplo ilustrado en la FIG. 5, una imagen 510 de una capa de base y una imagen 530 de una capa de mejora tienen el mismo tamaño de 64x64, pero la imagen 530 de la capa de mejora tiene una resolución que es cuatro veces la resolución de la imagen 510 de la capa de base.

Por consiguiente, para remitir a la imagen 510 de la capa de base en el momento de la predicción de la imagen 530 de la capa de mejora, se construye una imagen 520 de referencia sobremuestreando la imagen 510 de la capa de base reconstruida en el modo de predicción intra y, a continuación, la misma se usa para predecir la capa de mejora.

Las partes sombreadas en la imagen 530 de la capa de mejora indican bloques en los cuales no se aplica la predicción intercapa intra.

Cuando la predicción intercapa intra se aplica de esta manera, la imagen de la capa de base se puede someter a un reescalado.

Específicamente, el codificador de vídeo puede llevar a cabo un proceso de submuestreado de una imagen de entrada para codificar/descodificar la imagen de entrada según diversas capas que tienen resoluciones diferentes. El codificador de vídeo/descodificador de vídeo puede sobremuestrear una imagen reconstruida de una capa inferior para usar la imagen de la capa inferior como imagen de referencia en el transcurso de la codificación/descodificación.

Cuando se llevan a cabo el submuestreo y el sobremuestreo, una desadaptación en las características de fase puede provocar pérdidas en el proceso de codificación/descodificación y puede tener una influencia directa en el rendimiento de la codificación.

La FIG. 6 es un diagrama que ilustra esquemáticamente un ejemplo de reescalado (submuestreo/sobremuestreo) que se aplica en el transcurso de la predicción intercapa intra de acuerdo con la invención.

La FIG. 6(a) ilustra un ejemplo en el que, como muestra submuestreada, se usa una muestra entera en la misma posición. La FIG. 6(b) ilustra un ejemplo en el que no se usa una muestra entera en la misma posición sino que se usa una muestra que se desvía en 1/2 fase.

Tal como se ilustra en la FIG. 6(a) y la FIG. 6(b), cuando se aplica un filtro de sobremuestreo de medio píxel a muestras obtenidas por submuestreo de muestras originales, se pueden generar muestras en posiciones que se desvían en 1/2 fase con respecto a las posiciones de las muestreas submuestreadas.

En la FIG. 6(a) en la cual la fase no se desplaza en el momento del submuestreo, se generan muestras en las mismas posiciones que las muestras originales por sobremuestreo. Por el contrario, en la FIG. 6(b) en la cual la fase se desplaza en el momento del submuestreo, se generan muestras en posiciones (posiciones que se desvían en 1/2 fase) diferentes a las posiciones de las muestras originales. Por consiguiente, en la FIG. 6(b), puede producirse una pérdida entre las muestras originales y las muestras sobremuestreadas debido a la desadaptación de fase.

Para resolver este problema, se puede considerar que, en el proceso de sobremuestreo, se realiza un sobremuestreo desplazado en fase de adaptación de las muestras originales (imagen original) en fase.

La FIG. 7 es un diagrama que ilustra esquemáticamente un ejemplo del sobremuestreo desplazado en fase de acuerdo con la invención.

Las muestras inferiores en la FIG. 7 indican en conjunto muestras submuestradas con un desplazamiento de 1/2 fase con respecto a las muestras originales y muestras sobremuestreadas con respecto a las muestras submuestreadas.

En el ejemplo ilustrado en la FIG. 7, para compensar el desplazamiento de fase en el proceso de submuestreo, el sobremuestreo se lleva a cabo en posiciones de 1/4 fase y 3/4 fase sobre las muestras submuestreadas en el proceso de sobremuestreo.

Interpolando las muestras submuestreadas con el uso de las muestras de 1/4 fase y 3/4 fase, es posible eliminar la desadaptación de fase. En referencia a la FIG. 7, puede observarse que las muestras sobremuestreadas en las posiciones de 1/4 fase y 3/4 fase a partir de las muestras submuestreadas no presentan ninguna desadaptación de fase con las muestras originales.

En el momento de la aplicación del filtrado de sobremuestreo, el desplazamiento de fase para adaptarse a las muestras originales en fase se puede determinar en función del desplazamiento de fase usado en el momento del submuestreo.

Por ejemplo, para aplicar el filtrado de sobremuestreo en el descodificador de vídeo con el fin de no provocar una desadaptación de fase entre las muestras originales y las muestras sobremuestreadas, es necesario transmitir la información sobre el filtro de submuestreo aplicado por el codificador de vídeo o la información sobre el filtro de sobremuestreo a usar por el descodificador de vídeo, desde el codificador de vídeo al descodificador de vídeo.

La Tabla 1 muestra un ejemplo de información de filtro de submuestreo/sobremuestreo transmitida para la adaptación de fase de acuerdo con la invención.

<Tabla 1>

longitud de filtro: tamaño en tomas de filtro usado
Coeficiente del filtro: valor del coeficiente aplicado a cada toma
Fase desplazada: fase de la imagen submuestrada/sobremuestreada

En lugar de transmitir explícitamente la información del filtro desde el codificador de vídeo al descodificador de vídeo se puede usar una tabla de consulta.

En este momento, el codificador de vídeo puede transmitir un índice que indique la información del filtro en la tabla de consulta al descodificador de vídeo. El índice transmitido puede ser un índice que indique la información del filtro de submuestreo usado por el codificador de vídeo, o puede ser un índice que indique la información del filtro de sobremuestreo a usar por el descodificador de vídeo.

La Tabla 2 muestra un ejemplo de la tabla de consulta usada para transmitir la información de filtro de acuerdo con la invención.

<Tabla 2>

índice del filtro	descripción
00	**filtro de interpolación de medio píxel / 8 tomas**
01	**filtro de interpolación de medio píxel / 4 tomas**
10	**filtro de interpolación de 1/4, 3/4 fase / 8 tomas**
11	**filtro de interpolación de 1/4, 3/4 fase / 4 tomas**

La información del filtro de sobremuestreo/submuestreo se puede almacenar/transmitir en un nivel predeterminado del proceso de codificación/descodificación. Por ejemplo, la información del filtro se puede transmitir a través del uso de un conjunto de parámetros de secuencia. En este caso, se pueden usar los mismos filtros para la misma secuencia siempre que no se señalicen de forma diferente. La información del filtro se puede transmitir a través del uso de un conjunto de parámetros de imagen, y se pueden usar los mismos filtros para la misma imagen siempre que no se señalicen de manera diferente. Alternativamente, almacenando y transmitiendo la información del filtro a través del uso de un encabezamiento de franja (*slice*), se pueden usar los mismos filtros para la misma franja siempre que no se señalicen de manera diferente.

La Tabla 3 muestra brevemente un ejemplo en el que la información del filtro se transmite a través del uso de un conjunto de parámetros de secuencia.

<Tabla 3>

seq_parameter_set_rbsp( ) {	Descriptor
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
...	
adaptive_loop_filter_enabled_flag	u(1)
Interpolation_filter_indicator	u(2)
pcm_loop_filter_disable_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
temporal_id_nesting_flag	u(1)
rbsp_trailing_bits( )	
}	

En la Tabla 3, Interpolation_filter_indicator indica el tipo de un filtro de interpolación a utilizar.

La Tabla 3 muestra una estructura de sintaxis cuando la información del filtro se almacena en el conjunto de parámetros de secuencia, pero esto es solamente un ejemplo de la invención. El Interpolation_filter_indicator se puede transmitir a través del uso de un conjunto de parámetros de imagen o de un encabezamiento de franja según se ha descrito anteriormente.

El tipo del filtro indicado por el Interpolation_filter_indicator es información que indica la característica del filtro e incluye fase, tamaño en tomas, coeficiente de toma, y similares según se muestra en la Tabla 1.

Es decir, Interpolation_filter_indicator puede indicar un índice de la tabla de consulta según se muestra en la Tabla 4. La Tabla 4 muestra un ejemplo de una tabla usada para indicar la información de filtro aplicada al reescalado con el uso de Interpolation_filter_indicator.

<Tabla 4>

Interpolation_filter_indicator	descripción
00	filtro de interpolación de medio píxel / 8 tomas
01	filtro de interpolación de medio píxel / 4 tomas
10	filtro de interpolación de 1/4, 3/4 fase / 8 tomas
11	filtro de interpolación de 1/4, 3/4 fase / 4 tomas

La FIG. 8 es un diagrama que ilustra un ejemplo de un método que usa el Interpolation_filter_indicator de acuerdo con la invención, es decir, un método de muestreo cuando el valor de Interpolation_filter_indicator es 10.

La FIG. 8(a) ilustra un ejemplo de submuestreo con un desplazamiento de 1/2 fase aplicado en el mismo, y la FIG. 8(b) ilustra un ejemplo en el que el sobremuestreo se lleva a cabo sobre las muestras submuestreadas en la FIG. 8(a).

En la FIG. 8(b), con el fin de adaptar las fases entre las muestras originales y las muestras sobremuestreadas, se usa un filtro de interpolación de 8 tomas / 1/4 y 3/4 fase para llevar a cabo el sobremuestreo según se indica con el Interpolation_filter_indicator.

En otras palabras, el Interpolation_filter_indicator indica el filtro de sobremuestreo a usar por el descodificador de vídeo, e indica que se usará el “filtro de interpolación de 1/4 y 3/4 fase / 8 tomas” para adaptar la fase, por ejemplo, cuando el valor del Interpolation_filter_indicator es 10. El uso del filtro de interpolación de 1/4 y 3/4 fase / 8 tomas para adaptar la fase, significa que el codificador de vídeo ha realizado el submuestreo con un desplazamiento de 1/2 fase aplicado al mismo.

La predicción intercapa intra descrita hasta el momento la puede llevar a cabo el módulo de predicción intercapa (por ejemplo, el módulo de predicción de textura) descrito en referencia a las FIGS. 1 a 4. En la predicción intercapa intra, la textura del bloque de referencia en la capa de referencia se puede usar como valores predichos del bloque actual de la capa de mejora. En este caso, la textura del bloque de referencia se puede escalar por sobremuestreo.

El hecho de la aplicación de la predicción intercapa intra se puede señalizar en forma de una bandera subsiguiente a una bandera que indique si se va a dividir en particiones una CU. Cuando en la predicción intercapa intra se usa el escalado, se puede codificar y transmitir la información del filtro. En este momento, la información que se transmitirá es según se ha descrito anteriormente.

Predicción de movimiento intercapa

A la predicción de movimiento intercapa se le hace referencia también como predicción intercapa inter y, si es necesario, la predicción de movimiento intercapa y la predicción intercapa inter se pueden intercambiar con el fin de facilitar la interpretación de la invención en la presente descripción.

En la predicción de movimiento intercapa, el bloque actual de una capa actual (capa de mejora) se puede predecir usando la información de movimiento de una capa de referencia (capa de base).

La predicción de movimiento intercapa puede ser llevada a cabo por el módulo de predicción o el módulo de predicción intercapa ilustrados en las FIGS. 1 a 4. En lo sucesivo en la presente, se supone, por motivos de comodidad en la explicación, que la predicción de movimiento intercapa la lleva a cabo el módulo de predicción.

La FIG. 9 es un diagrama que ilustra brevemente un ejemplo de candidatos de información de movimiento que se usa para llevar a cabo la predicción inter en una capa sin referencia a otra capa (a lo que se hace referencia en lo sucesivo en la presente como “predicción inter”).

En la FIG. 9, A_0, A_1, B_0, B_1, B_2 y COL pueden indicar los bloques correspondientes o pueden indicar información de movimiento de los bloques correspondientes. En este caso, la información de movimiento de los bloques

correspondientes puede ser un vector de movimiento o puede ser un vector de movimiento y un índice de imagen de referencia.

En este caso, el método de predicción inter se describirá usando una capa de base como ejemplo.

La predicción inter en una capa de base puede ser llevada a cabo por el módulo de predicción o el módulo de predicción inter/intra ilustrados en las FIGS. 1 a 4. En lo sucesivo en la presente, por motivos de comodidad en la explicación, se supone que la predicción inter la lleva a cabo el módulo de predicción.

Los modos de predicción inter incluyen un modo de fusión, un modo de omisión, y un modo que usa un predictor de vectores de movimiento (MVP). Al modo que utiliza un MVP se le hace referencia también como modo AMVP (MVP avanzado) por motivos de comodidad en la explicación.

En el modo de fusión, como información de movimiento del bloque actual se puede la información de movimiento seleccionada a partir de la información de movimiento (a la que en lo sucesivo en la presente se hará referencia como candidatos de información de movimiento) de bloques vecinos ilustrados en la FIG. 9. La información que indica el candidato seleccionado de información de movimiento se puede transmitir desde el codificador de vídeo al descodificador de vídeo.

En el modo de omisión, como información de movimiento del bloque actual se usa la información de movimiento del candidato de información de movimiento seleccionado de la misma manera que en el modo de fusión, pero no se genera/transmite el valor residual.

Cuando se aplica el modo de fusión o el modo de omisión, el módulo de predicción puede determinar la disponibilidad de candidatos espaciales A_0, A_1, B_0, B_1, y B_2 en torno al bloque actual. La determinación de la disponibilidad se puede llevar a cabo en un orden predeterminado. Por ejemplo, la determinación de disponibilidad se puede llevar a cabo en el orden de $A_1 \rightarrow B_1 \rightarrow B_0 \rightarrow A_1 \rightarrow B_2$.

En este caso, la determinación de la disponibilidad de cada candidato puede incluir una determinación de igualdad con respecto al candidato previo. Por ejemplo, la disponibilidad de B_1 se puede realizar teniendo en cuenta si la información de movimiento es idéntica a la de A_1. Específicamente, cuando A_1 está disponible y A_1 y B_1 tienen la misma información de movimiento, se puede determinar que B_1 no está disponible.

De forma similar, la disponibilidad de B_0 se puede realizar teniendo en cuenta si la información de movimiento es idéntica a la de B_1, y la disponibilidad de A_0 se puede realizar teniendo en cuenta si la información de movimiento es idéntica a la de A_1.

La determinación de la disponibilidad de B_2 se puede llevar a cabo teniendo en cuenta tanto si B_2 tiene la misma información de movimiento que A_1 como si B_2 tiene la misma información de movimiento que B_1. En este momento, cuando está disponible la totalidad de los cuatro candidatos previos A_0, A_1, B_0, B_1, se puede determinar que B_2 no está disponible.

Cuando se usa el candidato COL, se puede especificar una imagen COL que incluye el candidato COL usando una lista de imágenes de referencia. Como candidato COL se puede la información de movimiento de un bloque de predicción que incluye una posición predeterminada en el bloque COL en la misma LCU que el bloque actual. En este momento, el vector de movimiento del candidato COL se puede escalar teniendo en cuenta la imagen COL y las imágenes de referencia de la imagen actual. El índice de referencia del candidato COL se puede fijar a un valor predeterminado (por ejemplo, 0).

Se puede construir una lista de candidatos de fusión de acuerdo con el orden de determinación de disponibilidad de los candidatos determinados como disponibles e incluyendo el candidato COL. En este momento, cuando el tipo de franja del bloque actual es B (es decir, una franja en la cual se aplica predicción bidireccional) y el número de candidatos incluidos en la lista de candidatos de fusión es menor que el número máximo, se pueden añadir candidatos (candidatos de bipredicción combinados) a la lista de candidatos de fusión.

Cuando el número de candidatos en la lista de candidatos de fusión es menor que el número máximo incluso después de haber construido tal como se ha descrito anteriormente la lista de candidatos de fusión, se puede añadir a la lista de candidatos de fusión un candidato predeterminado (por ejemplo, un candidato de fusión cero).

El módulo de predicción puede llevar a cabo la predicción inter usando la información de movimiento del candidato indicado por la información (por ejemplo, índice de fusión merge_idx) transmitida desde el codificador de vídeo en la lista de candidatos de fusión como información de movimiento del bloque actual. Por ejemplo, el módulo de predicción puede usar muestras indicadas por la información de movimiento de los candidatos seleccionados por el índice de fusión como muestras predichas del bloque actual.

Por otro lado, cuando se aplica el modo AMVP, el módulo de predicción también puede construir una lista de AMVP que incluye candidatos MVP.

En el modo AMVP, el módulo de predicción determina la disponibilidad de los candidatos en el orden de $A_0 \rightarrow A_1$, y determina la disponibilidad de candidatos en el orden de $B_0 \rightarrow B_1 \rightarrow B_2$.

A la hora de determinar la disponibilidad de candidatos en el orden de $A_0 \rightarrow A_1$, el módulo de predicción puede añadir al candidato correspondiente a la lista de AMVP (1) cuando un candidato que tiene la misma imagen de referencia que el bloque actual está presente como candidato disponible. Cuando no hay ningún candidato que cumple (1), el módulo de predicción puede escalar el vector de movimiento de un candidato que se ha determinado previamente como disponible (2) sobre la base de una diferencia de POC (Recuento de Orden de Imagen) entre una imagen actual y una imagen de referencia de la imagen actual y una diferencia de POC entre la imagen actual y una imagen de referencia del candidato. El módulo de predicción puede añadir el vector de movimiento escalado a la lista de AMVP.

Cuando la disponibilidad del candidato se determina en el orden de $B_0 \rightarrow B_1 \rightarrow B_2$, el módulo de predicción añade el candidato correspondiente a la lista de AMVP (1) cuando el candidato que tiene la misma imagen de referencia que el bloque actual está presente como candidato disponible. Cuando no hay ningún candidato que cumple (1) y no está disponible ninguno de A_0 y A_1, el módulo de predicción puede escalar el vector de movimiento de un candidato determinado previamente como disponible (2) sobre la base de una diferencia de POC entre una imagen actual y una imagen de referencia de la imagen actual y una diferencia de POC entre la imagen actual y una imagen de referencia del candidato. El módulo de predicción puede añadir el vector de movimiento escalado a la lista de AMVP.

Cuando se usa el candidato COL (candidato temporal), la imagen COL que incluye el candidato COL se puede especificar usando la lista de imágenes de referencia. Como candidato COL se puede usar la información de movimiento de un bloque de predicción que incluye una posición predeterminada en el bloque COL en la misma LCU que el bloque actual. En este momento, el vector de movimiento del candidato COL se puede escalar teniendo en cuenta la imagen COL y las imágenes de referencia de la imagen actual.

Cuando un candidato MVP determinado por medio de la determinación de disponibilidad de candidatos en el orden de $A_0 \rightarrow A_1$ es A, un candidato MVP determinado por medio de la determinación de disponibilidad de candidatos en el orden de $B_0 \rightarrow B_1 \rightarrow B_2$ es B, y un candidato MVP determinado por medio de la determinación de disponibilidad de candidatos temporales es COL, la lista de AMPV se puede construir en el orden de [A B COL].

En este momento, el módulo de predicción puede suprimir uno de A y B de la lista de AMVP cuando A y B son iguales.

El módulo de predicción puede ajustar el número de candidatos MVP en la lista AMVP a 2 cuando está disponible la totalidad de entre A, B y COL. Por ejemplo, el módulo de predicción puede construir la lista de AMVP de manera que incluya A y B y puede eliminar COL de la lista de AMVP.

El módulo de predicción puede añadir un vector de movimiento cero (0) como candidato cuando el número de candidatos en la lista de AMVP es menor que 2.

El codificador de vídeo puede transmitir hacia el descodificador de vídeo un índice de MVP que indica un MVP a utilizar en la predicción inter del bloque actual en la lista de AMVP, una diferencia de vector de movimiento mvd, y un índice de referencia que indica una imagen de referencia para el bloque actual en la lista de imágenes de referencia. La lista de imágenes de referencia es una lista de imágenes de referencia a utilizar en la predicción inter y se clasifica en L0 para la predicción hacia adelante y L1 para la predicción hacia atrás.

El módulo de predicción puede construir un bloque predicho del bloque actual sobre la base del MVP indicado por el índice de MVP, el vector de movimiento obtenido a partir de mvd, y la imagen de referencia indicada por el índice de referencia.

Cuando, para construir un bloque predicho, se aplica el modo de fusión/modo de omisión o el modo de AMVP, el módulo de predicción puede construir un bloque reconstruido del bloque actual sobre la base del bloque predicho y un valor residual. Cuando se aplica el modo de omisión, el valor residual no se transmite y, por lo tanto, el módulo de predicción puede usar el bloque predicho como bloque reconstruido.

El método de predicción inter se ha descrito usando como ejemplo una capa de base, pero la predicción inter se puede llevar a cabo en el mismo método que se ha descrito anteriormente cuando la predicción inter se realiza sobre la capa de mejora sin utilizar información de otra capa.

Cuando la predicción inter se realiza sobre la capa de base según se ha descrito anteriormente, la predicción de movimiento intercapa se puede llevar a cabo sobre la capa de mejora usando la información de movimiento de la capa de base.

La predicción de movimiento intercapa la pueden llevar a cabo los módulos de predicción del codificador de vídeo y del descodificador de vídeo.

La FIG. 10 es un diagrama de flujo que ilustra esquemáticamente un ejemplo de un método de ejecución de la predicción de movimiento intercapa de acuerdo con la invención.

En referencia a la FIG. 10, el módulo de predicción para una capa de mejora puede obtener la información de movimiento de una capa de referencia (S1010). Por ejemplo, el módulo de predicción intercapa para la capa de mejora puede obtener la información de movimiento de la capa de referencia sobre la base de la información transmitida desde el módulo de predicción para la capa de referencia. Alternativamente, el módulo de predicción para la capa de mejora puede obtener la información de movimiento de la capa de referencia sobre la base de información transmitida desde el codificador de vídeo.

En este momento, el módulo de predicción para la capa de mejora puede escalar la información de movimiento obtenida de la capa de referencia, por ejemplo, el vector de movimiento obtenido de la capa de referencia.

El módulo de predicción de la capa de mejora puede llevar a cabo la predicción intercapa inter sobre el bloque actual usando la información de movimiento de la capa de referencia (S1020). Por ejemplo, el módulo de predicción inter/intra para la capa de mejora puede predecir el bloque actual usando la información de movimiento de la capa de referencia obtenida por el módulo de predicción intercapa como candidato del modo de fusión/modo de omisión o el modo AMVP.

La predicción intercapa intra de acuerdo con la invención se describirá específicamente en referencia a los dibujos adjuntos.

1. Obtención de información de movimiento de la capa de referencia

La FIG. 11 es un diagrama que ilustra esquemáticamente un método de obtención de la información de movimiento de la capa de referencia de acuerdo con la invención. La FIG. 11 ilustra un ejemplo en el que la capa actual es una capa superior de una capa de referencia y la resolución de la capa actual es mayor que la resolución de la capa de referencia.

En referencia a la FIG. 11, una PU actual se puede especificar sobre la base de una PU 1100 (PU de referencia) de la capa de referencia en correspondencia con una PU 1110 (PU actual) de la capa actual.

Se supone que la posición para especificar la PU actual es (xCurr, yCurr) y la posición de la capa de referencia correspondiente a la PU actual, por ejemplo, la posición para especificar la PU de referencia es (xRef, yRef).

Se supone que el vector de movimiento intercapa que se debe obtener a partir del vector de movimiento de la capa de referencia es mvIL y el vector de movimiento de la capa de referencia (por ejemplo, el vector de movimiento de la PU de referencia) especificada por (xRef, yRef) es mvRL.

En la FIG. 11, nPSW es la anchura de la PU actual 1110 y nPSH es la altura de la PU actual 1110.

El módulo de predicción puede obtener la información de movimiento (por ejemplo, el vector de movimiento) de la PU de referencia especificando la PU actual y especificando la PU de referencia sobre la base de la posición de la PU actual.

(1) Determinación de la posición (xCurr, yCurr) para especificar la PU actual

La posición (xCurr, yCurr) para especificar la PU actual se puede determinar de manera que sea uno cualquiera de los candidatos ① a ⑫.

① LT = (xP, yP)

② RT = (xP+nPSW-1, yP)

③ LB = (xP, yP+nPSH-1)

④ RB = (xP+nPSW-1, yP+nPSH-1)

⑤ LT' = (xP-1, yP-1)

⑥ RT' = (xP+nPSW, yP-1)

⑦ LB' = (xP-1, yP+nPSH)

⑧ RB' = (xP+nPSW, yP+nPSH)

⑨ C0 = (xP+ (nPSW>>1)-1, yP+(nPSH>>1)-1)

⑩ C1 = (xP + (nPSW>>1), yP+(nPSH>>1)-1)

⑪ C2 = (xP+(nPSW>>1)-1, yP+ (nPSH>>1))

⑫ C3 = (xP+ (nPSW>>1), yP+(nPSH>>1))

La posición (xCurr, yCurr) para especificar la PU actual se puede determinar de manera que sea uno cualquiera de ① a ⑫ y se puede usar de forma fija, o se puede determinar a través de la RDO en el codificador de vídeo y, a continuación, se puede señalizar qué posición usar como (xCurr, yCurr).

Alternativamente, se puede determinar que la misma posición que se corresponde con la posición para especificar una PU en la capa de referencia (capa de base) es la posición para especificar una PU en la capa actual (capa de mejora). Por ejemplo, cuando, como posición para especificar la PU, se usa la esquina superior izquierda en la PU de la capa de referencia, la esquina superior izquierda LT=(xP, yP) en la PU de la capa actual se puede determinar como (xCurr, yCurr) destinada a usarse.

(2) Posición objetivo (xRef, yRef) en la capa de referencia

La posición (posición de la PU de referencia) de la cual se toma un vector de movimiento en la capa de referencia se puede determinar a partir de la posición de la PU actual en función de la relación entre la capa actual y la capa de referencia.

La Expresión 1 representa el método de determinación de la posición de la cual se toma el vector de movimiento en la capa de referencia de acuerdo con la invención.

<Expresión 1>

xRef = xCurr / escala

yRef = xCurr / escala

En este momento, el coeficiente escala que indica la relación de la capa actual con respecto ala capa de referencia se puede determinar en función de las resoluciones de las dos capas. Por ejemplo, cuando la resolución de la capa actual es el doble de la resolución de la capa de referencia, el valor de escala a aplicar es 2. Cuando la resolución de la capa actual es igual a la resolución de la capa de referencia, el valor de escala a aplicar es 1.

El valor de escala se determina de manera que es la relación de resolución de la capa actual con respecto a la capa de referencia, aunque la invención no se limita a la relación de resolución. El coeficiente escala se puede determinar en función del tipo de escalabilidad a aplicar entre la capa actual y la capa de referencia. por ejemplo, el coeficiente escala puede ser una relación de tamaño de imagen o una velocidad de frecuencia de cuadro entre la capa actual y la capa de referencia.

El módulo de predicción puede obtener el vector de movimiento en la posición (xRef, yRef), es decir, el vector de movimiento de una PU (PU de referencia) que cubre la posición (xRef, yRef), como mvRL.

El módulo de predicción puede obtener el índice de referencia de la PU (PU de referencia) que cubre la posición (xRef, yRef) como un índice de referencia refIdxIL a usar en la predicción de movimiento intercapa.

El módulo de predicción puede obtener el vector de movimiento mvIL a usar en la predicción de movimiento intercapa (predicción intercapa inter) escalando mvRL.

La Expresión 2 representa el método de obtención de mvIL escalando mvRL de acuerdo con la invención.

<Expresión 2>

mvIL = escala * mvRL

El coeficiente escala en la Expresión 2 representa la relación de la capa actual con respecto al a capa de referencia, de manera similar a la Expresión 1. Por ejemplo, cuando la resolución de la capa actual es el doble de la resolución de la capa de referencia, el valor de escala a aplicar es 2.

Cuando la resolución de la capa actual es igual a la resolución de la capa de referencia, el valor de escala a aplicar es 1 y el módulo de predicción puede usar mvRL como mvIL.

2. Predicción intercapa inter usando información de movimiento obtenida a partir de la capa de referencia

El módulo de predicción puede realizar la predicción intercapa inter sobre el bloque actual de la capa actual (capa de mejora) usando la información de movimiento obtenida a partir de la capa de referencia. La información de movimiento obtenida a partir de la capa de referencia incluye un vector de movimiento mvIL y un índice de referencia refIdxIL.

Por ejemplo, cuando se aplica el modo de fusión/modo de omisión, el módulo de predicción puede añadir mvIL y refIdxIL como candidatos de fusión a la lista de candidatos de fusión para el bloque actual.

Cuando se aplica el modo AMVP, el módulo de predicción puede añadir mvIL como candidato MVP a la lista de AMVP para el bloque actual.

(1) Cuando se aplica el modo de fusión

La Tabla 5 muestra un ejemplo de la lista de candidatos de fusión construida cuando no se remite a otra capa en la predicción inter y el modo de fusión se usa en una capa según se ha descrito anteriormente.

<Tabla 5>

Lista de candidatos de fusión de la predicción inter
- A_1 cuando A_1 está disponible
- B_1 cuando B_1 está disponible
- B_0 cuando B_0 está disponible
- A_0 cuando A_0 está disponible
- B_2 cuando no están disponibles A_1, B_1, B_0, A_0 pero está disponible B_2
- COL
- candidato de bipredicción combinado cuando no es mayor que el número máximo de candidatos
- candidato de vector de movimiento cero cuando no es mayor que el número máximo de candidatos

En la Tabla 5, A_1, B_1, B_0, A_0, B_2 y COL son iguales a A_1, B_1, B_0, A_0, B_2 y COL de la FIG. 9. El método de construcción de la lista de candidatos de fusión de la predicción inter en la Tabla 5 es el mismo que en el modo de fusión de la predicción inter descrita anteriormente en referencia a la FIG. 9.

En la Tabla 5, al candidato situado en la posición que está más arriba se le puede asignar un índice con el valor más pequeño y al candidato situado en la posición que está más abajo se le puede asignar un índice con el valor más grande.

Por el contrario, cuando se aplica la predicción de movimiento intercapa, el módulo de predicción puede construir la lista de candidatos de fusión incluyendo la información de movimiento obtenida a partir de la capa de referencia a diferencia de la Tabla 5. En este caso, por motivos de comodidad en la explicación, la información de movimiento obtenida a partir de la capa de referencia se indica como candidato de capa de referencia REF.

REF incluye mvIL y refIdxIL.

La Tabla 6 muestra un ejemplo de la lista de candidatos de fusión construida por el módulo de predicción cuando se aplica el modo de fusión de la predicción de movimiento intercapa de acuerdo con la invención. La Tabla 6 muestra el orden en el que se añaden candidatos de la capa de referencia a la lista de candidatos de fusión de acuerdo con la invención.

<Tabla 6>

Lista de candidatos de fusión de la predicción de movimiento intercapa
- ⓐ
- A_1 cuando A_1 está disponible
- B_1 cuando B_1 está disponible
- ⓑ

Lista de candidatos de fusión de la predicción de movimiento intercapa
- B_0 cuando B_0 está disponible - A_0 cuando A_0 está disponible - B_2 cuando no están disponibles A_1, B_1, B_0, A_0 pero está disponible B_2 - ⓒ - COL - ⓓ - candidato de bipredicción combinado cuando no es mayor que el número máximo de candidatos - ⓔ - candidato de vector de movimiento cero cuando no es mayor que el número máximo de candidatos - ⓕ

En la Tabla 6, A_1, B_1, B_0, A_0, B_2 y COL son iguales a A_1, B_1, B_0, A_0, B_2 y COL de la FIG. 9. El método de construcción de la lista de candidatos de fusión que hace uso de A_1, B_1, B_0, A_0, B_2 y COL en la Tabla 6 es el mismo que en el modo de fusión de la predicción inter descrita anteriormente en referencia a la FIG. 9.

En la Tabla 6, al candidato situado en la posición que está más arriba se le puede asignar un índice con el valor más pequeño y al candidato situado en la posición que está más abajo se le puede asignar un índice con el valor más grande.

En este caso, tal como se ha descrito en referencia a la FIG. 9, el módulo de predicción puede considerar la igualdad con un candidato previo siempre que se determine la disponibilidad de A_1, B_1, B_0, A_0 y B_2.

El módulo de predicción puede determinar la igualdad de A_1, B_1, B_0, A_0, B_2 y COL en un instante de tiempo después de construir la lista con la igualdad de A_1, B_1, B_0, A_0, y B_2 excluida. En este caso, la operación de dejar uno de los candidatos iguales se puede llevar a cabo después de que se determine la disponibilidad de COL.

Cuando se usa el modo de fusión de la predicción de movimiento intercapa, el módulo de predicción puede añadir el REF a cualquiera de ⓐ a ⓕ en la lista de candidatos de fusión de la Tabla 6.

Por ejemplo, el módulo de predicción puede asignar el índice más pequeño (por ejemplo, 0) al REF y puede añadir el REF a ⓐ en la lista de candidatos de fusión. Es decir, el módulo de predicción puede añadir el REF al inicio de la lista de candidatos de fusión.

El módulo de predicción puede asignar el índice más grande (por ejemplo, el número máximo de candidatos en la lista de candidatos de fusión -1) al REF y puede añadir el REF a ⓕ en la lista de candidatos de fusión. Es decir, el módulo de predicción puede añadir el REF al final de la lista de candidatos de fusión.

El módulo de predicción puede añadir el REF a ⓒ después de los candidatos espaciales. Es decir, el módulo de predicción puede añadir el REF a la lista de candidatos de fusión después de que se determine la disponibilidad de los candidatos espaciales.

El módulo de predicción puede añadir el REF a ⓓ después de los candidatos de unipredicción y antes del candidato de bipredicción combinado de entre los candidatos de la capa actual. Es decir, el módulo de predicción puede añadir REF a la lista de candidatos de fusión después de determinar la disponibilidad de los candidatos de unipredicción de entre los candidatos de la capa actual y antes de añadir el candidato de bipredicción combinado.

El módulo de predicción puede añadir el REF a ⓔ después de considerar todos los candidatos de la capa actual. Es decir, el módulo de predicción puede añadir el REF a la lista de candidatos de fusión después de comprobar la disponibilidad de todos los candidatos en la capa actual.

Además, el módulo de predicción puede añadir el REF a ⓑ después de considerar el candidato izquierdo y el candidato superior del bloque actual. Es decir, el módulo de predicción puede añadir el REF a la lista de candidatos de fusión después de comprobar secuencialmente la disponibilidad del candidato izquierdo y el candidato superior del bloque actual.

El número de candidatos incluidos en la lista de candidatos de fusión de la predicción de movimiento intercapa en la Tabla 6, es decir, el número máximo de candidatos, puede ser el mismo que en la Tabla 5. En este caso, para satisfacer el número máximo de candidatos, los candidatos de la capa actual situados después del REF se pueden excluir de la

lista de candidatos de fusión en función de la posición del REF. Cuando el número máximo de candidatos queda satisfecho por los candidatos de la capa actual situados antes del REF, el REF se puede excluir de la lista de candidatos de fusión.

El número de candidatos incluidos en la lista de candidatos de fusión de la predicción de movimiento intercapa en la Tabla 6, es decir, el número máximo de candidatos, puede ser diferente del correspondiente de la Tabla 5. Por ejemplo, el número máximo de candidatos en la Tabla 6 puede ser mayor en uno que el correspondiente de la Tabla 5 teniendo en cuenta el REF. En este caso, se puede considerar que la lista de candidatos de fusión se ha completado añadiendo el REF a una posición predeterminada o en un orden predeterminado después de que la lista de candidatos de fusión se haya construido con los candidatos de la capa actual. En este caso, la posición u orden del REF en la lista de candidatos de fusión se puede determinar de antemano o puede ser ordenada por el codificador de vídeo o puede ser obtenida por el descodificador de vídeo.

El codificador de vídeo puede ordenar qué candidato usar para realizar el modo de fusión de entre los candidatos de la lista de candidatos de fusión. Por ejemplo, el módulo de predicción puede seleccionar un candidato indicado por la información (por ejemplo, índice de fusión merge_idx) recibida desde el codificador de vídeo en la lista de candidatos de fusión mostrada en la Tabla 6 y puede construir un bloque predicho del bloque actual sobre la base del bloque indicado por la información de movimiento seleccionada.

(2) Cuando se aplica el modo MVP (AMVP)

La Tabla 7 muestra un ejemplo de una lista de candidatos construida cando no se remite a otra capa en la predicción inter y se aplica el modo MVP en la capa actual. En esta descripción, al modo de predicción inter que usa el MVP se le hace referencia como modo AMVP, y a una lista que incluye los MVPs candidatos usados en ese momento se le hace referencia como lista de AMPV.

<Tabla 7>

Lista de AMVP de predicción inter
- A seleccionado de entre A_0 y A_1
- B seleccionado de entre B_0, B_1, y B_2
- COL cuando no está disponible A o B
- candidato de vector de movimiento cero cuando no es mayor que el número máximo de candidatos

En la Tabla 7, A_1, B_1, B_0, A_0, B_2, COL, A y B son iguales a A_1, B_1, B_0, A_0, B_2, y COL de la FIG. 9, y a A y B en el modo MVP descrito en referencia a la FIG. 9. El método de construcción de la lista de candidatos de MVP de la predicción inter en la Tabla 7 es el mismo que en el modo MVP de la predicción inter descrita anteriormente en referencia a la FIG. 9.

En la Tabla 7, al candidato situado en la posición que está más arriba se le puede asignar un índice con el valor más pequeño y al candidato situado en la posición que está más abajo se le puede asignar un índice con el valor más grande.

Por el contrario, cuando se aplica la predicción de movimiento intercapa, el módulo de predicción puede construir una lista de candidatos de MVP incluyendo la información de movimiento obtenida a partir de la capa de referencia a diferencia de la Tabla 7. En este caso, al vector de movimiento obtenido a partir de la capa de referencia se le hace referencia como candidato de capa de referencia REF.

El REF incluye mvIL.

La Tabla 8 muestra un ejemplo de la lista de AMVP construida por el módulo de predicción cuando se aplica el modo MVP de la predicción de movimiento intercapa de acuerdo con la invención. La Tabla 8 muestra el orden de adición de los candidatos de la capa de referencia a la lista de AMVP de acuerdo con la invención.

<Tabla 8>

Lista de AMVP de predicción de movimiento intercapa
- ⓐ
- A seleccionado de entre A_0 y A_1
- ⓑ
- B seleccionado de entre B_0, B_1 y B_2
- ⓒ
- COL

- ⓓ - candidato de vector de movimiento cero cuando no es mayor que el número máximo de candidatos - ⓔ

En la Tabla 8, A_1, B_1, B_0, A_0, B_2, COL, A y B son iguales a A_1, B_1, B_0, A_0, B_2 y COL de la FIG. 9 y a A y B en el modo MVP descrito en referencia a la FIG. 9. El método de construcción de la lista de AMVP que hace uso de A, B, y COL en la Tabla 8 es el mismo que en el modo de fusión de la predicción inter descrita anteriormente en referencia a la FIG. 9.

En la Tabla 8, al candidato situado en la posición que está más arriba se le puede asignar un índice con el valor más pequeño y al candidato situado en la posición que está más abajo se le puede asignar un índice con el valor más grande.

El módulo de predicción puede excluir uno de A y B de la lista de AMVP cuando A y B son iguales entre sí como resultado de determinación de igualdad de A y B según se ha descrito en referencia a la FIG. 9. Este proceso se puede realizar en el momento de la determinación de la disponibilidad de B, se puede realizar después de la determinación de A y B, o se puede realizar cuando se determine la disponibilidad de COL o después de determinar la disponibilidad de COL.

Cuando se aplica el modo MVP de la predicción de movimiento intercapa, el módulo de predicción puede añadir el REF a cualquiera de ⓐ a ⓔ en la lista de AMVP mostrada en la Tabla 8.

Por ejemplo, el módulo de predicción puede asignar el índice más pequeño (por ejemplo, 0) al REF y puede añadir el REF a ⓐ en la lista de AMVP. Es decir, el módulo de predicción puede añadir el REF al inicio de la lista de AMVP.

El módulo de predicción puede asignar el índice más grande (por ejemplo, el número máximo de candidatos en la lista de AMVP -1) al REF y puede añadir el REF a ⓔ. Es decir, el módulo de predicción puede añadir el REF al final de la lista de AMVP.

El módulo de predicción puede añadir el REF a ⓓ, que es la posición después de considerar todos los candidatos de la capa actual.

El módulo de predicción puede añadir el REF a ⓒ después de los candidatos espaciales. El módulo de predicción puede añadir el REF a ⓑ, que es la posición después de considerar el candidato izquierdo del bloque actual y antes de considerar el candidato superior.

El número de candidatos incluidos en la lista de AMVP de la predicción de movimiento intercapa en la Tabla 8, es decir, el número máximo de candidatos, puede ser el mismo que en la Tabla 7. En este caso, para satisfacer el número máximo de candidatos, los candidatos de la capa actual situados después del REF se pueden excluir de la lista de AMVP en función de la posición del REF. Cuando el número máximo de candidatos queda satisfecho por los candidatos de la capa actual situados antes del REF, el REF se puede excluir de la lista de AMVP.

Por ejemplo, cuando el número máximo de candidatos es 2, el REF en la posición de ⓒ se puede excluir de la lista de AMVP.

El número de candidatos incluidos en la lista de AMVP de la predicción de movimiento intercapa en la Tabla 8, es decir, el número máximo de candidatos, puede ser diferente del de la Tabla 7. Por ejemplo, el número máximo de candidatos en la Tabla 8 puede ser mayor en uno que el correspondiente de la Tabla 7 (por ejemplo, 2) teniendo en cuenta el REF. En este caso, se puede considerar que la lista de AMVP se ha completado añadiendo el REF a una posición predeterminada después de que la lista de AMVP se haya construido con los candidatos de la capa actual. En este caso, la posición u orden del REF en la lista de AMVP se puede determinar de antemano o puede ser ordenada por el codificador de vídeo o puede ser obtenida por el descodificador de vídeo.

El codificador de vídeo puede dar instrucciones sobre qué candidato usar para ejecutar el modo MVP de entre los candidatos de la lista de AMVP. Por ejemplo, el módulo de predicción puede seleccionar un candidato indicado por la información recibida del codificador de vídeo en la lista de AMVP mostrada en la Tabla 8 y puede obtener el vector de movimiento del bloque actual usando el vector de movimiento seleccionado y el mvd recibido del codificador de vídeo. El módulo de predicción puede construir un bloque predicho del bloque actual sobre la base del vector de movimiento obtenido y de la imagen de referencia indicada por el índice de referencia recibido desde el codificador de vídeo.

Por otro lado, cuando se cumple una condición predeterminada, el módulo de predicción puede escalar mvIL antes de añadir el REF a la lista de AMVP.

Por ejemplo, cuando el índice de referencia de una PU actual es diferente del índice de referencia (refIdxIL) de una PU de referencia, el mvIL se puede escalar. En otras palabras, cuando el POC de una imagen (una imagen de referencia de la PU actual) indicada por el índice de referencia de la PU actual es diferente del POC de una imagen (una imagen de referencia de la PU de referencia) indicada por el índice de referencia de la PU de referencia, el mvIL se puede escalar.

La FIG. 12 es un diagrama que ilustra esquemáticamente el método de escalado del mvIL de acuerdo con la invención.

En referencia a la FIG. 12, la diferencia entre el POC de una imagen actual en una capa actual y el POC de una imagen 1220 de referencia de una PU actual 1200 es tb, y la diferencia entre el POC de la imagen actual en una capa de referencia y el POC de una imagen 1230 de referencia de una PU 1210 de referencia es td.

Cuando el POC pocRef de la imagen 1220 de referencia de la PU actual 1200 es diferente del POC pocRefLayer de la imagen 1230 de referencia de la PU 1210 de referencia, el módulo de predicción puede escalar el REF, es decir, mvIL, y puede añadir el REF escalado a la lista de AMVP.

En este caso, el mvIL se puede escalar usando un método enunciado con la Expresión 3. Al mvIL escalado se le hace referencia como mvIL'.

<Expresión 3>

tx = (16384+(Abs(td)>>1))/td

DistScaleFactor = Clip3(-4096, 4095, (tb*tx+32)>>6)

mvIL' = Clip3(-8192, 8191,75, Sign(DistScaleFactor*mvIL)*((Abs(DistScaleFactor*mvIL)+127)>>8))

td = Clip3(-128, 127, pocCurr-pocRefLayer)

tb = Clip3(-128, 127, pocCurr-pocRef)

En la Expresión 3, pocCurr representa el POC de la imagen actual, pocRef representa el POC de una imagen indicada por el índice de referencia de la PU actual, y pocRefLayer representa el POC de una imagen indicada por el índice de referencia a la PU de referencia, es decir, el índice de referencia de (xRef, yRef).

Abs(x) es igual a -x cuando x es menor que 0, y es igual a x cuando x es igual o superior a 0. Clip3(x, y, z) es igual a x cuando z es menor que x, es igual a y cuando z es mayor que y, y es igual a z en caso contrario.

Cuando pocRef y pocRefLayer son diferentes entre sí, el módulo de predicción puede escalar el vector de movimiento intercapa sobre la base de la distancia a la imagen de referencia en cada capa según expone la Expresión 3, y puede añadir el candidato de vector de movimiento escalado (es decir, mvIL escalado o REF escalado) a la lista de AMVP.

En este momento, el módulo de predicción puede añadir el REF escalado (es decir, mvIL escalado) en lugar del REF (es decir, mvIL) a la lista de AMVP, y puede ejecutar la predicción de movimiento intercapa usando el modo AMVP de la misma manera que la descrita anteriormente.

Cuando se realiza la predicción de movimiento intercapa, se pueden usar todas las particiones de CU con independencia del tamaño de bloque de la capa de base desde el punto de vista del nivel de CU. Desde el punto de vista del nivel de PU, el codificador de vídeo aplica la RDO sobre la predicción inter y la predicción de movimiento intercapa, con lo cual es posible aplicar un modo de predicción óptimo.

Predicción de sintaxis intercapa

En la predicción de sintaxis intercapa, se predice o genera una textura de un bloque actual usando información de sintaxis de una capa de referencia. En este momento, la información de sintaxis de la capa de referencia usada para predecir el bloque actual puede ser información sobre el modo de predicción intra o información de movimiento.

Por ejemplo, una capa de referencia puede ser una franja P o una franja B, pero el bloque de referencia en la franja puede ser un bloque en el cual se ha aplicado el modo de predicción intra. En este caso, se puede ejecutar la predicción intercapa de generación/predicción de la textura de la capa actual usando el modo intra de entre la información de sintaxis de la capa de referencia. Específicamente, cuando la capa de referencia es una franja P o una franja B pero el bloque de referencia en la franja es un bloque en el cual se ha aplicado el modo de predicción intra, la predicción intra

se puede realizar (1) en el modo de predicción intra del bloque de referencia (2) usando los píxeles de referencia vecinos del bloque actual en la capa actual mediante la aplicación de la predicción de sintaxis intercapa.

Cuando la capa de referencia es una franja P o una franja B pero el bloque de referencia en la franja es un bloque en el cual se ha aplicado el modo de predicción inter, la predicción intercapa de generación/predicción de la textura de la capa actual mediante escalado de la información de movimiento de entre la información de sintaxis de la capa de referencia se puede realizar de la misma manera que en la predicción de movimiento intercapa antes descrita.

Por consiguiente, la predicción de sintaxis intercapa puede ser un método de utilización de la predicción de movimiento intercapa y la predicción de textura intercapa juntas.

Tal como se ha descrito anteriormente, la predicción de movimiento intercapa es un método de predicción que genera una señal de predicción (bloque predicho) en la capa de mejora usando la información de movimiento de la capa de referencia.

En este caso, la información de movimiento de la capa de referencia se puede escalar en función de la variación de la resolución entre capas.

Cuando se aplica el modo de predicción intra a la capa de referencia y un bloque sin información de movimiento es un bloque de referencia, es posible generar la señal de predicción tomando el modo de predicción intra de la capa de referencia según se ha descrito anteriormente y prediciendo el bloque actual a partir de los píxeles vecinos de la capa de mejora.

En la predicción de sintaxis intercapa o la información de movimiento intercapa, la información de movimiento de componentes de luma se puede tomar en las unidades de bloques de 4x4 a partir de la capa de referencia. En este caso, la información de movimiento de componentes de luma se puede usar como información de movimiento de componentes de croma.

Cuando se lleva a cabo la predicción de sintaxis intercapa, se pueden aplicar de manera adaptativa la predicción intercapa intra y la predicción intercapa de movimiento en función del modo de predicción aplicado al bloque de referencia en la capa de referencia señalizando qué modo de predicción aplicar en un nivel superior sin generar un modo de PU nuevo desde el punto de vista del nivel de PU.

Una bandera que indique si aplicar la predicción de sintaxis intercapa se puede transmitir en forma de una bandera de manera posterior a una bandera (por ejemplo, CU_split_flag) que indique la división de una CU.

La FIG. 13 es un diagrama que ilustra brevemente un ejemplo del método de ejecución de la predicción de sintaxis intercapa de acuerdo con la invención.

En referencia a la FIG. 13, en un bloque 1310 de referencia de una capa 1300 de referencia hay presencia de bloques intra y bloques inter.

Cuando la predicción de sintaxis intercapa se aplica al ejemplo ilustrado en la FIG. 13, un bloque reconstruido se puede construir construyendo una imagen 1320 sobremuestreada de manera que se adapte a la capa actual y aplicando a la misma la predicción de sintaxis intercapa (1330).

En este momento, la predicción de sintaxis intercapa se puede realizar por emisión al modo de predicción intra a partir de los bloques (intra) en los cuales se aplica el modo de predicción intra en la capa de referencia y por remisión a la información de movimiento de los bloques (MV) a los cuales se aplica el modo de predicción inter.

La predicción de sintaxis intercapa para una imagen objetivo de la capa actual se puede combinar con otros modos de predicción. Las regiones rayadas en una imagen actual 1340 de la FIG. 13 representan regiones en las que se pueden aplicar los otros modos de predicción.

Predicción residual intercapa

La predicción residual intercapa es un método de generación de un vídeo predicho residual de una capa de mejora usando una señal residual de una capa de referencia y codificando/descodificando el vídeo residual en referencia a un vídeo predicho residual en la capa de mejora.

La predicción residual intercapa la puede llevar a cabo el módulo de predicción o el módulo de predicción intercapa de las FIGS. 1 a 4. Por motivos de comodidad en la explicación, se supone que la predicción residual intercapa la lleva a cabo el módulo de predicción.

El módulo de predicción puede generar un vídeo predicho residual de la capa de mejora escalando la señal residual de la capa de referencia en función de una diferencia o relación de resolución entre capas.

El codificador de vídeo puede aplicar la RDO en la predicción residual intercapa de manera independiente con respecto a la predicción de sintaxis intercapa y la predicción intercapa intra. Cuando se determina, a través de la RDO, que se lleva a cabo la predicción residual intercapa, una bandera para predecir una señal residual, es decir, una bandera que indique que se lleva a cabo la predicción residual intercapa, se puede codificar y transmitir antes del valor residual (coeficientes de transformada) en las unidades de CU.

La FIG. 14 es un diagrama que ilustra esquemáticamente un método de ejecución de la predicción residual intercapa de acuerdo con la invención.

En referencia a la FIG. 14, un vídeo predicho residual 1420 se puede generar escalando una parte 1410 de bloque de referencia a la que se hará referencia en una señal residual 1400 de una capa de referencia para la predicción intercapa en función de la relación de resolución entre capas.

Un vídeo residual 1430 correspondiente al bloque actual se puede reconstruir sobre la base del vídeo predicho residual 1420. En este momento, las regiones rayadas representan regiones en las cuales no se aplica la predicción residual intercapa.

El codificador de vídeo puede determinar si aplicar la predicción residual intercapa a las señales residuales en todos los modos de predicción sobre la base de la RDO. La información que indica si aplicar la predicción residual intercapa se puede transmitir en forma de una bandera antes de transmitir la información sobre los coeficientes (por ejemplo, los coeficientes de transformada cuantificados del valor residual).

Estructura de sintaxis para la predicción intercapa

El codificador de vídeo puede señalizar información requerida para la predicción intercapa antes mencionada. El descodificador de vídeo puede recibir la información señalizada y puede llevar a cabo la predicción intercapa.

Se describirá a continuación el método de realización de la predicción intercapa antes descrita o que se describirá posteriormente, y ejemplos de sintaxis que indican información necesaria.

La Tabla 9 muestra un ejemplo de sintaxis de una unidad de NAL de acuerdo con la invención.

<Tabla 9>

nal_unit(NumBytesInNALunit) {	**Descriptor**
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
nal_unit_type	u(6)
NumBytesInRBSP = 0	
nalUnitHeaderBytes = 1	
if(nal_unit_type == 1 || nal_unit_type == 4 || nal_unit_type == 5	
|| nal_unit_type == SVC_NAL)	
temporal_id	u(3)
output_flag	u(1)
reserved_one_4bits	u(4)
if(nal_unit_type == SVC_NAL)	
nal_unit_header_svc_extension( )	
nalUnitHeaderBytes += 2	
}	
for(i = nalUnitHeaderBytes; i < NumBytesInNALunit; i++) {	
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRBSP++]	b(8)
rbsp_byte[NumBytesInRBSP++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++]	b(8)
}	
}	

La Tabla 10 muestra un ejemplo de sintaxis de extensión SVC de un encabezamiento de unidad de NAL de acuerdo con la invención.

<Tabla 10>

nal_unit_header_svc_extension( ) {	Descriptor
idr_flag	u(1)
dependency_id	u(3)
quality_id	u(4)
}	

La Tabla 11 muestra un ejemplo de sintaxis RBSP de un conjunto de parámetros de secuencia de acuerdo con la invención.

<Tabla 11>

seq_parameter_set_rbsp( ) {	Descriptor
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
[Ed. (BB): Not in HM, further discuss separate_colour_plane_flag]	
max_temporal_layers_minus1	u(3)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
[Ed. (BB): chroma bit depth present in HM software but not used further]	
pcm_enabled_flag	u(1)
if (pcm_enabled_flag) {	
pcm_bit_depth_luma_minus1	u(4)
pcm_bit_depth_chroma_minus1	u(4)
}	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
max_num_ref_frames	ue(v)
num_reorder_frames	ue(v)
max_dec_frame_buffering	ue(v)
max_latency_increase	ue(v)
log2_min_coding_block_size_minus3	ue(v)
log2_diff_max_min_coding_block_size	ue(v)
log2_min_transform_block_size_minus2	ue(v)
log2_diff_max_min_transform_block_size	ue(v)
if (pcm_enabled_flag) {	
log2_min_pcm_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_coding_block_size	ue(v)
}	
max_transform_hierarchy_depth_inter	ue(v)

max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enable_flag	
chroma_pred_from_luma_enabled_flag	u(1)
deblocking_filter_in_APS_enabled_flag	u(1)
loop_filter_across_slice_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
adaptive_loop_filter_enabled_flag	u(1)
if (pcm_enabled_flag)	
pcm_loop_filter_disable_flag	u(1)
temporal_id_nesting_flag	u(1)
[Ed. (BB): x y padding syntax missing here, present in HM software]	
if (log2_min_coding_block_size_minus3 = = 0)	
inter_4x4_enabled_flag	u(1)
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
if (num_tile_columns_minus1 != 0 || num_tile_rows_minus1 != 0) {	
uniform_spacing_flag	u(1)
if (!uniform_spacing_flag) {	
for (i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for (i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
tile_boundary_independence_flag	u(1)
if (tile_boundary_independence_flag == 1)	
loop_filter_across_tile_flag	u(1)
}	
sps_extension_flag	u(1)
if(sps_extension_flag)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

La Tabla 12 muestra un ejemplo de sintaxis RBSP de un subconjunto de parámetros de secuencia de acuerdo con la invención.

<Tabla 12>

subset_seq_parameter_set_rbsp( ) {	**Descriptor**
seq_parameter_set_rbsp( )	
if(profile_idc == SVC_PROFILE) {	
seq_parameter_set_svc_extension( )	
}	
}	

La Tabla 13 muestra un ejemplo de sintaxis de extensión SVC de un conjunto de parámetros de secuencia de acuerdo con la invención.

<Tabla 13>

seq_parameter_set_svc_extension( ) {	**Descriptor**
singleloop_decoding_flag	u(1)
}	

La Tabla 14 muestra un ejemplo de sintaxis RBSP de un conjunto de parámetros de imagen de acuerdo con la invención.

<Tabla 14>

pic_parameter_set_rbsp( ) {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
num_short_term_ref_pic_sets	ue(v)
for(idx = 0; idx < num_short_term_ref_pic_sets; idx++)	
short_term_ref_pic_set(idx)	
long_term_ref_pics_present_flag	u(1)
entropy_coding_synchro	u(v)
cabac_istate_reset_flag	u(1)
if(entropy_coding_synchro)	
num_substreams_minus1	ue(v)
num_temporal_layer_switching_point_flags	ue(v)
for(i = 0; i < num_temporal_layer_switching_point_flags; i++)	
temporal_layer_switching_point_flag[i]	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
[Ed. (BB): not present in HM software]	
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
enable_temporal_mvp_flag	u(1)
slice_granularity	u(2)
max_cu_qp_delta_depth	ue(v)
chroma_cb_qp_offset	se(v)
chroma_cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
tile_info_present_flag	u(1)
tile_control_present_flag	u(1)

if(tile_info_present_flag = = 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
if(num_tile_columns_minus1 != 0 || num_tile_rows_minus1 != 0) {	
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
}	
}	
if (tile_control_present_flag) {	
if (num_tile_columns_minus1 != 0 || num_tile_rows_minus1 != 0) {	
tile_boundary_independence_flag	u(1)
if (tile_boundary_independence_flag == 1)	
loop_filter_across_tile_flag	u(1)
}	
}	
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data( ))	
pps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

La Tabla 15 muestra un ejemplo de sintaxis de datos de lista de escalado de acuerdo con la invención.

<Tabla 15>

scaling_list_param( ) {	Descriptor
scaling_list_present_flag	u(1)
if(scaling_list_present_flag)	
for(SizeID = 0; SizeID < 4; SizeID++)	
for(MatrixID = 0; MatrixID < (SizeID = = 3) ? 2 : 6; MatrixID++) {	
scaling_list_pred_mode_flag	u(1)
if(!scaling_list_pred_mode_flag)	
scaling_list_pred_matrix_id_delta	ue(v)
else	
scaling_list(ScalingList[SizeID][MatrixID][0], (1 << (4 + (sizeID << 1))))	
}	
}	

La Tabla 16 muestra un ejemplo de sintaxis de lista de escalado de acuerdo con la invención.

<Tabla 16>

scaling_list(ScalingList, coefNum) {	Descriptor
nextcoef = 8	u(1)
for(i=0; i < coefNum, i++) {	
scaling_list_delta_coef	se(v)
nextcoef = (nextcoef + scaling_list_delta_coef + 256) % 256	
ScalingList[i] = nextcoef	
}	
}	

La Tabla 17 muestra un ejemplo de sintaxis RBSP de un conjunto de parámetros adaptativo de acuerdo con la invención.

<Tabla 17>

aps_rbsp( ) {	Descriptor
aps_id	ue(v)
aps_scaling_list_data_present_flag	u(1)
if(aps_scaling_list_data_present_flag)	
scaling_list_param( )	
aps_deblocking_filter_flag	u(1)
if (aps_deblocking_filter_flag) {	
disable_deblocking_filter_flag	u(1)
if (!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
aps_sample_adaptive_offset_flag	u(1)
if(aps_sample_adaptive_offset_flag)	
sao_param( )	
aps_adaptive_loop_filter_flag	u(1)
if(aps_adaptive_loop_filter_flag)	
alf_param( )	
aps_extension_flag	u(1)
if(aps_extension_flag)	
while(more_rbsp_data( ))	
aps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

La Tabla 18 muestra un ejemplo de sintaxis RBSP de información de mejora complementaria de acuerdo con la invención.

<Tabla 18>

sei_rbsp( ) {	Descriptor
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

La Tabla 19 muestra la sintaxis de un mensaje de información de mejora complementaria de acuerdo con la invención.

<Tabla 19>

sei_message( ) {	Descriptor
payloadType = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
sei_payload(payloadType, payloadSize)	
}	

La Tabla 20 muestra un ejemplo de sintaxis RBSP de un delimitador de unidades de acceso de acuerdo con la invención.

<Tabla 20>

access_unit_delimiter_rbsp( ) {	Descriptor
primary_pic_type	u(3)
rbsp_trailing_bits( )	
}	

La Tabla 21 muestra un ejemplo de sintaxis RBSP de datos de relleno de acuerdo con la invención.

<Tabla 21>

filler_data_rbsp( ) {	Descriptor
while(next_bits(8) == 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

La Tabla 22 muestra un ejemplo de sintaxis RBSP de una capa de franjas de acuerdo con la invención.

<Tabla 22>

slice_layer_rbsp( ) {	Descriptor
slice_header( )	
slice_data( )	
rbsp_slice_trailing_bits( )	
}	

La Tabla 23 muestra un ejemplo de sintaxis RBSP de una extensión de capa de franjas de acuerdo con la invención.

<Tabla 23>

slice_layer_extension_rbsp( ) {	Descriptor
slice_header_in_scalable_extension( )	
slice_data_in_scalable_extension( )	
}	

La Tabla 24 muestra un ejemplo de sintaxis de bits finales (*trailing bits*) de franja RBSP de acuerdo con la invención.

<Tabla 24>

rbsp_slice_trailing_bits( ) {	Descriptor
rbsp_trailing_bits( )	
while(more_rbsp_trailing_data( ))	
cabac_zero_word /* equal to 0x0000 */	f(16)
}	

La Tabla 25 muestra un ejemplo de sintaxis de bits finales RBSP de acuerdo con la invención.

<Tabla 25>

rbsp_trailing_bits( ) {	Descriptor
rbsp_stop_one_bit /* equal to 1 */	f(1)
while(!byte_aligned( ))	
rbsp_alignment_zero_bit /* equal to 0 */	f(1)
}	

La Tabla 26 muestra un ejemplo de sintaxis de alineación de bytes RBSP de acuerdo con la invención.

<Tabla 26>

byte_align( ) {	Descriptor
while(!byte_aligned( ))	
bit_equal_to_one	f(1)
}	

La Tabla 27 muestra un ejemplo de un encabezamiento de franja de acuerdo con la invención.

<Tabla 27>

slice_header( ) {	Descriptor
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag) {	
pic_parameter_set_id	ue(v)
if(IdrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
}	
else {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_pps_flag	u(1)
if(!short_term_ref_pic_set_pps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lt_minus1[i]	ue(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
}	
}	
if(scaling_list_enable_flag || deblocking_filter_in_APS_enabled_flag || sample_adaptive_offset_enabled_flag || adaptive_loop_filter_enabled_flag) {	
if(sample_adaptive_offset_enabled_flag)	
slice_sample_adaptive_offset_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	
slice_adaptive_loop_filter_flag	u(1)
aps_id	ue(v)
}	
if(slice_type = = P || slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
ref_pic_list_modification( )	
ref_pic_list_combination( )	
}	
if(slice_type != I)	
cabac_init_idc	ue(v)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
inherit_dbl_params_from_APS_flag	u(1)
if (!inherit_dbl_params_from_APS_flag) {	
disable_deblocking_filter_flag	u(1)
if (!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_idc = = 1 && slice_type = = B))	
pred_weight_table( )	
}	
if(slice_type = = P || slice_type = = B)	
5_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag && aps_adaptive_loop_filter_flag)	
alf_cu_control_param( )	
for(i = 0; i < num_substreams_minus1 + 1; i++){	
substream_length_mode	u(2)
substream_length[i]	u(v)
}	
}	

La Tabla 28 muestra un ejemplo de encabezamiento de franja en sintaxis de extensión escalable de acuerdo con la invención.

<Tabla 28>

slice_header_in_scalable_extension( ) {	Descriptor
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag) {	
pic_parameter_set_id	ue(v)
if(IdrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
}	
else {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_pps_flag	u(1)
if(!short_term_ref_pic_set_pps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lt_minus1[i]	ue(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
}	
}	
if(scaling_list_enable_flag || deblocking_filter_in_APS_enabled_flag || sample_adaptive_offset_enabled_flag || adaptive_loop_filter_enabled_flag) {	
if(sample_adaptive_offset_enabled_flag)	
slice_sample_adaptive_offset_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	

slice_adaptive_loop_filter_flag	u(1)
aps_id	ue(v)
}	
if(slice_type = = EP || slice_type = = EB) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = EB)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
ref_pic_list_modification( )	
ref_pic_list_combination( )	
}	
if (singleloop_decoding_flag) {	
if (slice_type == EI) //Syntax related inter-layer prediction	
inter_layer_intra_prediction_flag	u(1)
inter_layer_differential_coding_flag	u(1)
else {	
inter_layer_syntax_prediction_flag	u(1)
inter_layer_residual_prediction_flag	u(1)
}	
}	
else {	
inter_layer_intra_prediction_flag	u(1)
if (slice_type != EI) {	
inter_layer_residual_prediction_flag	u(1)
}	
inter_layer_differential_coding_flag	u(1)
}	
if(slice_type != EI)	
cabac_init_idc	ue(v)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
inherit_dbl_params_from_APS_flag	u(1)
if (!inherit_dbl_params_from_APS_flag) {	
disable_deblocking_filter_flag	u(1)
if (!disable_deblocking_filter_flag) {	

beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
if(slice_type = = EB)	
collocated_from_l0_flag	u(1)
if((weighted_pred_flag && slice_type = = EP) || (weighted_bipred_idc = = 1 && slice_type = = EB))	
pred_weight_table( )	
}	
if(slice_type = = EP | | slice_type = = EB)	
5_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag && aps_adaptive_loop_filter_flag)	
alf_cu_control_param( )	
for(i = 0; i < num_substreams_minus1 + 1; i++){	
substream_length_mode	u(2)
substream_length[i]	u(v)
}	
}	

La Tabla 29 muestra un ejemplo de sintaxis de un conjunto de imágenes de referencia de corto plazo de acuerdo con la invención.

<Tabla 29>

short_term_ref_pic_set(idx) {	**Descriptor**
inter_ref_pic_set_prediction_flag	u(1)
if(inter_ref_pic_set_prediction_flag) {	
delta_idx_minus1	ue(v)
delta_rps_sign	u(1)
abs_delta_rps_minus1	ue(v)
for(j = 0; j <= NumDeltaPocs[RIdx]; j++) {	
ref_idc0[j]	u(1)
if(!**ref_idc0**[j])	
ref_idc1[j]	u(1)
}	
}	
else {	
num_negative_pics	ue(v)
num_positive_pics	ue(v)
for(i = 0; i < num_negative_pics; i++) {	
delta_poc_s0_minus1[i]	ue(v)
used_by_curr_pic_s0_flag[i]	u(1)
}	
for(i = 0; i < num_positive_pics; i++) {	
delta_poc_s1_minus1[i]	ue(v)
used_by_curr_pic_s1_flag[i]	u(1)
}	
}	
}	

La Tabla 30 muestra un ejemplo de sintaxis de modificación de lista de imágenes de referencia de acuerdo con la invención.

<Tabla 30>

ref_pic_list_modification( ) {	Descriptor
if(slice_type == P || slice_type == B || slice_type == EP || slice_type == EB) {	
ref_pic_list_modification_flag_l0	u(1)
if(ref_pic_list_modification_flag_l0)	
do {	
ref_pic_list_modification_idc	ue(v)
if(ref_pic_list_modification_idc != 3)	
ref_pic_set_idx	ue(v)
} while(ref_pic_list_modification_idc != 3)	
}	
if(slice_type == B || slice_type == EB) {	
ref_pic_list_modification_flag_l1	u(1)
if(ref_pic_list_modification_flag_l1)	
do {	
ref_pic_list_modification_idc	ue(v)
if(ref_pic_list_modification_idc != 3)	
ref_pic_set_idx	ue(v)
} while(ref_pic_list_modification_idc != 3)	
}	
}	

La Tabla 31 muestra un ejemplo de sintaxis de combinación de listas de imágenes de referencia de acuerdo con la invención.

<Tabla 31>

ref_pic_list_combination( ) {	Descriptor
if(slice_type == B || slice_type == EB) {	
ref_pic_list_combination_flag	u(1)
if(ref_pic_list_combination_flag) {	
num_ref_idx lc_active_minus1	ue(v)
ref_pic_list_modification_flag_lc	u(1)
if(ref_pic_list_modification_flag_lc)	
for (i =0; i <= num_ref_idx_lc_active_minus1; i++) {	
pic_from_list_0_flag	u(1)
ref_idx_list_curr	ue(v)
}	
}	
}	
}	

La Tabla 32 muestra un ejemplo de sintaxis de un parámetro de compensación adaptativa según muestras, de acuerdo con la invención.

<Tabla 32>

sao_param( ) {	Descriptor
sao_split_param(0, 0, 0, 0)	
sao_offset_param(0, 0, 0, 0)	
sao_flag_cb	u(1)
if(sao_flag_cb) {	
sao_split_param(0, 0, 0, 1)	
sao_split_param(0, 0, 0, 1)	
}	
sao_flag_cr	u(1)
if(sao_flag_cr) {	
sao_split_param(0, 0, 0, 2)	
sao_split_param(0, 0, 0, 2)	
}	
}	

sao_split_param(rx, ry, saoDepth , cIdx) {	Descriptor
if(saoDepth < SaoMaxDepth)	
sao_split_flag[cIdx][saoDepth][rx][ry]	u(1)
else	
sao_split_flag[cIdx][saoDepth][rx][ry] = 0	
if(sao_split_flag[cIdx][saoDepth][rx][ry]) {	
sao_split_param(2*rx + 0, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_split_param(2*rx + 1, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_split_param(2*rx + 0, 2*ry + 1, saoDepth + 1 , cIdx)	
sao_split_param(2*rx + 1, 2*ry + 1, saoDepth + 1 , cIdx)	
}	
}	

sao_offset_param(rx, ry, saoDepth , cIdx) {	Descriptor
if(sao_split_flag[cIdx][saoDepth][rx][ry]) {	
sao_offset_param(2*rx + 0, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_offset_param(2*rx + 1, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_offset_param(2*rx + 0, 2*ry + 1, saoDepth + 1 , cIdx)	
sao_offset_param(2*rx + 1, 2*ry + 1, saoDepth + 1 , cIdx)	
} else {	
sao_type_idx[cIdx][saoDepth][rx][ry]	ue(v)
if(sao_type_idx[cIdx][saoDepth][rx][ry] != 0)	
for(i = 0, i < NumSaoClass[sao_type_idx], i++)	
sao_offset[cIdx][saoDepth][x0][y0][i]	se(v)
}	
}	

La Tabla 33 muestra un ejemplo de sintaxis de un parámetro de filtro de bucle adaptativo de acuerdo con la invención.

<Tabla 33>

alf_param() {	Descriptor
alf_region_adaptation_flag	u(1)
alf_length_luma_minus_5_div2	ue(v)
alf_no_filters_minus1	ue(v)
if (alf_no_filters_minus1 == 1)	
alf_start_second_filter	ue(v)
else if (alf_no_filters_minus1 > 1) {	
for (i=1; i< (alf_region_adaptation_flag ? 16 : 15); i++)	
alf_filter_pattern[i]	u(1)
}	
if (AlfNumFilters > 1)	
alf_pred_method	u(1)
for (i=0; i< AlfNumFilters; i++)	
alf_nb_pred_luma[i]	u(1)
for (i=0; i< AlfNumFilters; i++)	
for (j=0; j< AlfCodedLengthLuma; j++)	
alf_coeff_luma[i][j]	ge(v)
alf_chroma_idc	ue(v)
if (alf_chroma_idc) {	
alf_length_chroma_minus_5_div2	ue(v)
for(i = 0; i< AlfCodedLengthChroma; i++)	
alf_coeff_chroma[i]	se(v)
}	
}	

La Tabla 34 muestra un ejemplo de sintaxis de un parámetro de control de unidades de codificación del filtro de bucle adaptativo de acuerdo con la invención.

<Tabla 34>

alf_cu_control_param() {	Descriptor
alf_cu_control_flag	u(1)
if(alf_cu_control_flag) {	
alf_cu_control_max_depth	ue(v)
alf_length_cu_control_info	se(v)
for(i = 0; i < NumAlfCuFlag; i++)	
alf_cu_flag[i]	u(1)
}	
}	

La Tabla 35 muestra un ejemplo de sintaxis de una tabla de pesos de predicción de acuerdo con la invención.

<Tabla 35>

pred_weight_table( ) {	Descriptor
luma_log2_weight_denom	ue(v)
if(chroma_format_idc != 0)	
delta_chroma_log2_weight_denom	se(v)
if (slice_type == P || (slice_type == B && ref_pic_list_combination_flag == 0)) {	
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {	
luma_weight_l0_flag	u(1)
if(luma_weight_l0_flag) {	
delta_luma_weight_l0[i]	se(v)
luma_offset_l0[i]	se(v)
}	
if (chroma_format_idc != 0) {	
chroma_weight_l0_flag	u(1)
if(chroma_weight_l0_flag)	
for(j =0; j < 2; j++) {	
delta_chroma_weight_l0[i][j]	se(v)
delta_chroma_offset_l0[i][j]	se(v)
}	
}	
}	
}	
if(slice_type == B) {	
if (ref_pic_list_combination_flag == 0) {	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {	
luma_weight_l1_flag	u(1)
if(luma_weight_l1_flag) {	
delta_luma_weight_l1[i]	se(v)
luma_offset_l1[i]	se(v)
}	
if(chroma_format_idc != 0) {	
chroma_weight_l1_flag	u(1)
if(chroma_weight_l1_flag)	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l1[i][j]	se(v)
delta_chroma_offset_l1[i][j]	se(v)
}	
}	
}	
} else {	
for(i = 0; i <= num_ref_idx_lc_active_minus1; i++) {	
luma_weight_lc_flag	u(1)
if(luma_weight_l1_flag) {	
delta_luma_weight_lc[i]	se(v)
luma_offset_lc[i]	se(v)
}	
if(chroma_format_idc != 0) {	
chroma_weight_lc_flag	u(1)
if(chroma_weight_lc_flag)	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_lc[i][j]	se(v)
delta_chroma_offset_lc[i][j]	se(v)
}	
}	
}	
}	
}	

La Tabla 36 muestra un ejemplo de sintaxis de datos de franja de acuerdo con la invención.

<Tabla 36>

slice_data( ) {	Descriptor
CurrTbAddr = LCUAddress	
moreDataFlag = 1	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlagIdx = -1	
do {	
XLCU = HorLumaLocation(CurrTbAddr)	
YLCU = VerLumaLocation(CurrTbAddr)	
moreDataFlag = coding_tree(XLCU, YLCU, Log2TbSize, 0)	
CurrTbAddr = NextTbAddress(CurrTbAddr)	
if(CurrTbAddr = = firstTbInTileAddr)	
rbsp_trailingbits()	
} while(moreDataFlag)	
}	

La Tabla 37 muestra un ejemplo de datos de franja en sintaxis de extensión escalable de acuerdo con la invención.

<Tabla 37>

slice_data_in_scalable_extension ( ) {	Descriptor
CurrTbAddr = LCUAddress	
moreDataFlag = 1	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlagIdx = -1	
do {	
XLCU = HorLumaLocation(CurrTbAddr)	
YLCU = VerLumaLocation(CurrTbAddr)	
moreDataFlag = coding_tree_in_scalale_extension (XLCU, YLCU, Log2TbSize, 0)	
CurrTbAddr = NextTbAddress(CurrTbAddr)	
if(CurrTbAddr = = firstTbInTileAddr)	
rbsp_trailingbits()	
} while(moreDataFlag)	
}	

La Tabla 38 muestra un ejemplo de sintaxis de un árbol de codificación de acuerdo con la invención.

<Tabla 38>

coding_tree(x0, y0, log2CUSize, cuDepth) {	Descriptor
if(x0 + (1 << log2CUSize) <= $PicWidthInSamples_L$ && y0 + (1 << log2CUSize) <= $PicHeightInSamples_L$ && cuAddress(x0, y0) >= SliceAddress && log2CUSize > Log2MinCUSize) {	
split_coding_unit_flag[x0][y0]	ae(v)
}	
if(adaptive_loop_filter_flag && alf_cu_control_flag) {	
if(cuDepth <= alf_cu_control_max_depth)	
if(cuDepth == alf_cu_control_max_depth ||	
split_coding_unit_flag[x0][y0] == 0)	
AlfCuFlagIdx++	
}	
if(cu_qp_delta_enabled_flag && log2CUSize >= log2MinCUDQPSize)	
IsCuQpDeltaCoded = 0	
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
if(cuAddress(x1, y0) > SliceAddress)	
moreDataFlag = coding_tree(x0, y0, log2CUSize – 1, cuDepth + 1)	
if(cuAddress(x0, y1) > SliceAddress && moreDataFlag && x1 < $PicWidthInSamples_L$)	
moreDataFlag = coding_tree(x1, y0, log2CUSize – 1, cuDepth + 1)	
if(cuAddress(x1, y1) > SliceAddress && moreDataFlag && y1 < $PicHeightInSamples_L$)	
moreDataFlag = coding_tree(x0, y1, log2CUSize – 1, cuDepth + 1)	
if(moreDataFlag && x1 < $PicWidthInSamples_L$ && y1 < $PicHeightInSamples_L$)	
moreDataFlag = coding_tree(x1, y1, log2CUSize – 1, cuDepth + 1)	
} else {	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlag[x0][y0] = alf_cu_flag[AlfCuFlagIdx]	
coding_unit(x0, y0, log2CUSize)	
if(granularity_block_boundary(x0, y0, log2CUSize)) {	
end_of_slice_flag	ae(v)

moreDataFlag = !end_of_slice_flag	
} else	
moreDataFlag = 1	
}	
return moreDataFlag	
}	

La Tabla 39 muestra un ejemplo de árbol de codificación en sintaxis de extensión escalable de acuerdo con la invención.

<Tabla 39>

coding_tree_in_scalable_extension (x0, y0, log2CUSize, cuDepth) {	**Descriptor**
if(x0 + (1 << log2CUSize) <= PicWidthInSamples$_L$ && y0 + (1 << log2CUSize) <= PicHeightInSamples$_L$ && cuAddress(x0, y0) >= SliceAddress && log2CUSize > Log2MinCUSize) {	
split_coding_unit_flag[x0][y0]	ae(v)
}	
if(adaptive_loop_filter_flag && alf_cu_control_flag) {	
if(cuDepth <= alf_cu_control_max_depth)	
if(cuDepth == alf_cu_control_max_depth ||	
split_coding_unit_flag[x0][y0] == 0)	
AlfCuFlagIdx++	
}	
if(cu_qp_delta_enabled_flag && log2CUSize >= log2MinCUDQPSize)	
IsCuQpDeltaCoded = 0	
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
if(cuAddress(x1, y0) > SliceAddress)	
moreDataFlag = coding_tree_in_scalable_extension (x0, y0, log2CUSize – 1, cuDepth + 1)	
if(cuAddress(x0, y1) > SliceAddress && moreDataFlag && x1 < PicWidthInSamples$_L$)	
moreDataFlag = coding_tree_in_scalable_extension (x1, y0, log2CUSize – 1, cuDepth + 1)	
if(cuAddress(x1, y1) > SliceAddress && moreDataFlag && y1 < PicHeightInSamples$_L$)	
moreDataFlag = coding_tree_in_scalable_extension (x0, y1, log2CUSize – 1, cuDepth + 1)	
if(moreDataFlag && x1 < PicWidthInSamples$_L$ && y1 < PicHeightInSamples$_L$)	
moreDataFlag = coding_tree_in_scalable_extension (x1, y1, log2CUSize – 1, cuDepth + 1)	
} else {	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlag[x0][y0] = alf_cu_flag[AlfCuFlagIdx]	
coding_unit_in_scalable_extension (x0, y0, log2CUSize)	
if(granularity_block_boundary(x0, y0, log2CUSize)) {	
end_of_slice_flag	ae(v)
moreDataFlag = !end_of_slice_flag	
} else	
moreDataFlag = 1	
}	
return moreDataFlag	
}	

La Tabla 40 muestra un ejemplo de sintaxis de una unidad de codificación de acuerdo con la invención.

<Tabla 40>

coding_unit(x0, y0, log2CUSize) {	Descriptor
if(slice_type != I)	
skip_flag[x0 **][** y0]	ae(v)
if(skip_flag[x0][y0])	
prediction_unit(x0, y0 , log2CUSize)	
else if(slice_type != I || log2CUSize == Log2MinCUSize) {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(PredMode != MODE_INTRA || log2CUSize = = Log2MinCUSize)	
part_mode	ae(v)
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
x2 = x1 − ((1 << log2CUSize) >> 2)	
y2 = y1 − ((1 << log2CUSize) >> 2)	
x3 = x1 + ((1 << log2CUSize) >> 2)	
y3 = y1 + ((1 << log2CUSize) >> 2)	
if(PartMode == PART_2Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
} else if(PartMode == PART_2NxnU) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y2 , log2CUSize)	
} else if(PartMode == PART_2NxnD) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y3 , log2CUSize)	
} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x2, y0 , log2CUSize)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x3, y0 , log2CUSize)	
} else { /* PART_NxN */	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
prediction_unit(x1, y1 , log2CUSize)	
}	
if(!pcm_flag) {	
transform_tree(x0, y0, log2CUSize, log2CUSize, log2CUSize, 0, 0)	
transform_coeff(x0, y0, x0, y0, log2CUSize, log2CUSize, 0, 0)	
}	
}	
}	

La Tabla 41 muestra un ejemplo de unidad de codificación en sintaxis de extensión escalable de acuerdo con la invención.

<Tabla 41>

coding_unit_in_scalable_extension (x0, y0, log2CUSize) {	**Descriptor**
if(ILDiffCodingFlag)	
il_diff_mode[x0][y0]	
if(slice_type != EI)	
skip_flag[x0 **][** y0 **]**	ae(v)
if ((ILIntraPredFlag || ILSyntaxPredFlag)	
&& !skip_flag[x0][y0] && !il_diff_mode[x0][y0])	
il_mode[x0][y0]	ae(v)
if (!il_mode[x0][y0]){	
if(skip_flag[x0][y0])	
prediction_unit(x0, y0 , log2CUSize)	
else if(slice_type != EI || log2CUSize == Log2MinCUSize) {	
if(slice_type != EI)	
pred_mode_flag	ae(v)
if(PredMode != MODE_INTRA || log2CUSize == Log2MinCUSize)	
part_mode	ae(v)
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
x2 = x1 − ((1 << log2CUSize) >> 2)	
y2 = y1 − ((1 << log2CUSize) >> 2)	
x3 = x1 + ((1 << log2CUSize) >> 2)	
y3 = y1 + ((1 << log2CUSize) >> 2)	
if(PartMode == PART_2Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
} else if(PartMode == PART_2NxnU) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y2 , log2CUSize)	
} else if(PartMode == PART_2NxnD) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y3 , log2CUSize)	
} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x2, y0 , log2CUSize)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x3, y0 , log2CUSize)	
} else { /* PART_NxN */	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
prediction_unit(x1, y1 , log2CUSize)	
}	
}	
}	
if(ILResPredFlag && !ILIntraPredFlag && !il_diff_mode[x0][y0])	
il_res_mode[x0][y0]	ae(v)
if(!pcm_flag && !skip_flag[x0][y0]) {	
transform_tree(x0, y0, log2CUSize, log2CUSize, log2CUSize, 0, 0)	
transform_coeff(x0, y0, x0, y0, log2CUSize, log2CUSize, 0, 0)	
}	

La Tabla 42 muestra un ejemplo de sintaxis de una unidad de predicción de acuerdo con la invención.

<Tabla 42>

prediction_unit(x0, y0, log2CUSize) {	Descriptor
if(skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else if(PredMode == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && log2CUSize >= Log2MinIPCMCUSize && log2CUSize <= Log2MaxIPCMCUSize)	
pcm_flag	ae(v)
if(pcm_flag) {	
while (!byte_aligned( ))	
pcm_alignment_zero_bit	u(v)
for(i = 0; i < 1 << (log2CUSize << 1); i++)	
pcm_sample_luma[i **]**	u(v)
for(i = 0; i < (1 << (log2CUSize << 1)) >> 1; i++)	
pcm_sample_chroma[i **]**	u(v)
} else {	
prev_intra_luma_pred_flag[x0 **][** y0 **]**	ae(v)
if(prev_intra_luma_pred_flag[x0][y0])	
mpm_flag[x0 **][** y0 **]**	ae(v)
else	
rem_intra_luma_pred_mode[x0 **][** y0 **]**	ae(v)
intra_chroma_pred_mode[x0 **][** y0 **]**	ae(v)
SignaledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[x0][y0] == 3 : intra_chroma_pred_mode[x0][y0] == 2)	
}	
} else { /* MODE_INTER */	
merge_flag[x0 **][** y0 **]**	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else {	
if(slice_type == B || slice_type == EB)	
inter_pred_flag[x0 **][** y0 **]**	ae(v)
if(inter_pred_flag[x0][y0] == Pred_LC) {	
if(num_ref_idx_lc_active_minus1 > 0)	
ref_idx_lc[x0 **][** y0 **]**	ae(v)

mvd_coding(mvd_lc[x0][y0][0], mvd_lc[x0][y0][1])	
mvp_lc_flag[x0][y0]	ae(v)
}	
else { /* Pred_L0 or Pred_BI */	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0 **][** y0 **]**	ae(v)
mvd_coding(mvd_l0[x0][y0][0], mvd_l0[x0][y0][1])	
mvp_l0_flag[x0][y0]	ae(v)
}	
if(inter_pred_flag[x0][y0] == Pred_BI) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0 **][** y0 **]**	ae(v)
mvd_coding(mvd_l1[x0][y0][0], mvd_l1[x0][y0][1])	
mvp_l1_flag[x0][y0]	ae(v)
}	
}	
}	
}	

La Tabla 43 muestra un ejemplo de sintaxis de codificación diferencial por vectores de movimiento de acuerdo con la invención.

<Tabla 43>

mvd_coding(mvd_x, mvd_y) {	Descriptor
abs_mvd_greater0_flag[0]	ae(v)
abs_mvd_greater0_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0])	
abs_mvd_greater1_flag[0]	ae(v)
if(abs_mvd_greater0_flag[1])	
abs_mvd_greater1_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0]) {	
if(abs_mvd_greater1_flag[0])	
abs_mvd_minus2[0]	ae(v)
mvd_sign_flag[0]	ae(v)
}	
if(abs_mvd_greater0_flag[1]) {	
if(abs_mvd_greater1_flag[1])	
abs_mvd_minus2[1]	ae(v)
mvd_sign_flag[1]	ae(v)
}	
mvd_x = abs_mvd_greater0_flag[0] * (abs_mvd_minus2[0] + 2) * (1 – 2 * mvd_sign_flag[0])	
mvd_y = abs_mvd_greater0_flag[1] * (abs_mvd_minus2[1] + 2) * (1 – 2 * mvd_sign_flag[1])	
}	

La Tabla 44 muestra un ejemplo de sintaxis de un árbol de transformada de acuerdo con la invención.

<Tabla 44>

transform_tree(x0, y0, log2CUSize, log2TrafoWidth, log2TrafoHeight, trafoDepth, blkIdx) {	Descriptor
if(trafoDepth == 0 && IntraSplitFlag == 0 && PredMode != MODE_INTRA && !(PartMode == PART_2Nx2N && merge_flag[x0][y0]))	
no_residual_data_flag	ae(v)
if(!no_residual_data_flag) {	
log2TrafoSize = (log2TrafoWidth + log2TrafoHeight) >> 1	
intraSplitFlag = (IntraSplitFlag && trafoDepth == 0 ? 1 : 0)	
interSplitFlag = (max_transform_hierarchy_depth_inter = =0 && PredMode = = MODE_INTER && PartMode != PART_2Nx2N && trafoDepth = = 0)	
maxDepth = (PredMode == MODE_INTRA ? max_transform_hierarchy_depth_intra + IntraSplitFlag : max_transform_hierarchy_depth_inter + InterSplitFlag)	
xBase = x0 – (x0 & (1 << log2TrafoWidth))	
yBase = y0 – (y0 & (1 << log2TrafoHeight))	
if(log2TrafoSize <= Log2MaxTrafoSize && log2TrafoSize > Log2MinTrafoSize && trafoDepth < maxDepth && !intraSplitFlag && !interSplitFlag)	
split_transform_flag[x0][y0][trafoDepth]	ae(v)
if(PredMode != MODE_INTRA && log2TrafoSize <= Log2MaxTrafoSize) {	
firstChromaCbf = (log2TrafoSize = = Log2MaxTrafoSize || trafoDepth = = 0) ? 1 : 0	
if(firstChromaCbf || log2TrafoSize > Log2MinTrafoSize) { [Ed. (WJ): Log2MinTrafoSize or 2?]	
if(firstChromaCbf || cbf_cb[xBase][yBase][trafoDepth – 1]) {	
readCbf = true	
if(blkIdx = = 3 && log2TrafoSize < Log2MaxTrafoSize)	
readCbf = cbf_cb[xBase][yBase][trafoDepth] || cbf_cb[xBase + (1 << log2TrafoWidth)][yBase][trafoDepth] || cbf_cb[xBase][yBase + (1 << log2TrafoHeight)][trafoDepth]	
if (!readCbf)	
cbf_cb[x0][y0][trafoDepth] = 1	
else	
cbf_cb[x0][y0][trafoDepth]	ae(v)
}	
if(firstChromaCbf || cbf_cr[xBase][yBase][trafoDepth – 1]) {	
readCbf = true	
if(blkIdx = = 3 && log2TrafoSize < Log2MaxTrafoSize)	

readCbf = cbf_cr[xBase][yBase][trafoDepth] || cbf_cr[xBase + (1 << log2TrafoWidth)][yBase][trafoDepth] || cbf_cr[xBase][yBase + (1 << log2TrafoHeight)][trafoDepth]	
if (!readCbf)	
cbf_cr[x0][y0][trafoDepth] = 1	
else	
cbf_cr[x0][y0][trafoDepth]	ae(v)
}	
}	
}	
if(split_transform_flag[x0][y0][trafoDepth]) {	
if(InterTUSplitDirection == 2) {	
x1 = x0 + ((1 << log2TrafoWidth) >> 1)	
y1 = y0	
x2 = x0	
y2 = y0 + ((1 << log2TrafoHeight) >> 1)	
x3 = x1	
y3 = y2	
} else {	
x1 = x0 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y1 = y0 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection)	
x2 = x1 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y2 = y1 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection)	
x3 = x2 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y3 = y2 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection)	
log2TrafoHeight = log2TrafoHeight + 2 * InterTUSplitDirection – 1	
log2TrafoWidth = log2TrafoWidth – 2 * InterTUSplitDirection + 1	
}	
transform_tree(x0, y0, log2CUSize, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 0)	
transform_tree(x1, y1, log2CUSize, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 1)	
transform_tree(x2, y2, log2CUSize, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 2)	
transform_tree(x3, y3, log2CUSize, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 3)	
} else {	
if(PredMode == MODE_INTRA || trafoDepth != 0 || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) {	

readCbf = true	
if(blkIdx = = 3 && PredMode != MODE_INTRA && (log2CUSize <= Log2MaxTrafoSize+1) || (log2TrafoSize < Log2MaxTrafoSize))	
readCbf = cbf_luma[xBase][yBase][trafoDepth] || cbf_luma[xBase + (1 << log2TrafoWidth)][yBase][trafoDepth] || cbf_luma[xBase][yBase + (1 << log2TrafoHeight)][trafoDepth] || cbf_cb[xBase][yBase][trafoDepth – 1] || cbf_cr[xBase][yBase][trafoDepth – 1]	
if (!readCbf)	
cbf_luma[x0][y0][trafoDepth] = 1	
else	
cbf_luma[x0][y0][trafoDepth]	ae(v)
}	
if(PredMode = = MODE_INTRA)	
if(log2TrafoSize > Log2MinTrafoSize) { [Ed. (WJ): Log2MinTrafoSize or 2?]	
cbf_cb[x0][y0][trafoDepth]	ae(v)
cbf_cr[x0][y0][trafoDepth]	ae(v)
} else if(blkIdx = = 0) {	
cbf_cb[x0][y0][trafoDepth – 1]	ae(v)
cbf_cr[x0][y0][trafoDepth – 1]	ae(v)
}	
}	
}	
}	

La Tabla 45 muestra un ejemplo de sintaxis de un coeficiente de transformada de acuerdo con la invención.

<Tabla 45>

transform_coeff(x0, y0, xC, yC, log2TrafoWidth, log2TrafoHeight, trafoDepth, blkIdx) {	**Descriptor**
if(cbf_luma[x0][y0][trafoDepth] || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta	ae(v)
IsCuQpDeltaCoded = 1	
}	
if(split_transform_flag[x0][y0][trafoDepth]) {	
if(InterTUSplitDirection == 2) {	
x1 = x0 + ((1 << log2TrafoWidth) >> 1)	
y1 = y0	
x2 = x0	
y2 = y0 + ((1 << log2TrafoHeight) >> 1)	
x3 = x1	
y3 = y2	
} else {	
x1 = x0 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y1 = y0 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection)	
x2 = x1 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y2 = y1 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection)	
x3 = x2 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y3 = y2 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection)	
log2TrafoHeight = log2TrafoHeight + 2 * InterTUSplitDirection – 1	
log2TrafoWidth = log2TrafoWidth – 2 * InterTUSplitDirection + 1	
}	
transform_coeff(x0, y0, x0, y0, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 0)	
transform_coeff(x1, y1, x0, y0, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 1)	
transform_coeff(x2, y2, x0, y0, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 2)	
transform_coeff(x3, y3, x0, y0, log2TrafoWidth – 1, log2TrafoHeight – 1, trafoDepth + 1, 3)	
} else {	
log2TrafoSize = ((log2TrafoWidth + log2TrafoHeight) >> 1)	
log2TrafoSizeC = ((log2TrafoSize = = Log2MinTrafoSizeC) ? log2TrafoSize : log2TrafoSize – 1)	
if (PredMode = = MODE_INTRA) {	
scanIdx = ScanType[log2TrafoSize – 2][IntraPredMode]	
scanIdxC = ScanType[log2TrafoSize – 2][IntraPredModeC]	
} else {	
scanIdx = 0	
scanIdxC = 0	
}	
if (cbf_luma[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoWidth, log2TrafoHeight, scanIdx, 0)	
if (log2TrafoSize > Log2MinTrafoSize) { [Ed. (WJ): Log2MinTrafoSize or 2?]	
if (cbf_cb[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSizeC, trafoDepth, scanIdxC, 1)	
if (cbf_cr[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSizeC, trafoDepth, scanIdxC, 2)	
} else if (blkIdx = = 3) {	
if (cbf_cb[x0][y0][trafoDepth])	
residual_coding(xC, yC, log2TrafoSizeC, trafoDepth, scanIdxC, 1)	
if (cbf_cr[x0][y0][trafoDepth])	
residual_coding(xC, yC, log2TrafoSizeC, trafoDepth, scanIdxC, 2)	
}	
}	
}	
}	

La Tabla 46 muestra un ejemplo de sintaxis de codificación residual de acuerdo con la invención.

<Tabla 46>

residual_coding (x0, y0, log2TrafoWidth, log2TrafoHeight, scanIdx, cIdx) {	**Descriptor**
last_significant_coeff_x_prefix	ae(v)
last_significant_coeff_y_prefix	ae(v)
if (last_significant_coeff_x_prefix > 3)	
last_significant_coeff_x_suffix	ae(v)
if (last_significant_coeff_y_prefix > 3)	
last_significant_coeff_y_suffix	ae(v)
numCoeff = 0	
do {	
xC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][numCoeff][0]	
yC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][numCoeff][1]	
numCoeff++	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
numLastSubset = (numCoeff − 1) >> 4	
for(i = numLastSubset; i >= 0; i− −) {	
offset = i << 4	
if(max(log2TrafoWidth, log2TrafoHeight) > 3) {	
xCG = ScanOrder[log2TrafoWidth − 2][log2TrafoHeight − 2][scanIdx][i][0]	
yCG = ScanOrder[log2TrafoWidth − 2][log2TrafoHeight − 2][scanIdx][i][1]	
rightCGFlag = (xCG = = (1<< (log2TrafoWidth − 2)) − 1) ? 0 : significant_coeff_group_flag[xCG + 1][yCG]	
bottomCGFlag = (yCG = = (1 << (log2TrafoHeight − 2)) − 1) ? 0 : significant_coeff_group_flag[xCG][yCG + 1]	
if((i < numLastSubset) && (rightCGFlag + bottomCGFlag < 2) && (i > 0))	
significant_coeff_group_flag[xCG][yCG]	ae(v)
for(n = 15; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][0]	
yC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][1]	
if ((n + offset) < (numCoeff − 1) && significant_coeff_group_flag[xCG][yCG]) {	
numNZInCG = (i = = numLastSubset) ? 1 : 0	
if ((n > 0) || (rightCGFlag + bottomCGFlag = = 2) || (i = = 0) || (numNZInCG > 0)) {	
significant_coeff_flag[xC][yC]	ae(v)
numNZInCG += significant_coeff_flag[xC][yC]	
else	
significant_coeff_flag[xC][yC] = 1	
}	

}	
} else {	
for(n = 15; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][0]	
yC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][1]	
if((n + offset) < (numCoeff − 1))	
significant_coeff_flag[xC][yC]	ae(v)
}	
}	
for(n = 15; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][0]	
yC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][1]	
if(significant_coeff_flag[xC][yC])	
coeff_abs_level_greater1_flag[n]	ae(v)
}	
for(n = 15; n >= 0; n− −) {	
if(coeff_abs_level_greater1_flag[n])	
coeff_abs_level_greater2_flag[n]	ae(v)
}	
for(n = 15; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][0]	
yC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][1]	
if(significant_coeff_flag[xC][yC]) {	
coeff_sign_flag[n]	ae(v)
}	
for(n = 15; n >= 0; n− −) {	
if(coeff_abs_level_greater2_flag[n])	
coeff_abs_level_minus3[n]	ae(v)
xC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][0]	
yC = ScanOrder[log2TrafoWidth][log2TrafoHeight][scanIdx][n + offset][1]	
if(significant_coeff_flag[xC][yC]) {	
transCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_minus3[n] + 3) * (1 − 2 * coeff_sign_flag[n])	
} else	
transCoeffLevel[x0][y0][cIdx][xC][yC] = 0	
}	
}	
}	

En la sintaxis antes mencionada, **nal_unit_type** se puede definir como en la Tabla 47.

<Tabla 47>

nal_unit_type	Contenido de unidad de NAL y estructura de sintaxis RBSP	Clase de tipo de unidad de NAL
0	Sin especificar	no VCL
1	Franja codificada de una imagen no IDR y no CRA slice_layer_rbsp ( )	VCL
2-3	Reservado	n/a
4	Franja codificada de una imagen CRA slice_layer_rbsp ( )	VCL
5	Franja codificada de una imagen IDR slice_layer_rbsp ( )	VCL
6	Información de mejora complementaria (SEI) sei_rbsp ( )	no VCL
7	Conjunto de parámetros de secuencia seq_parameter_set_rbsp ( )	no VCL
8	Conjunto de parámetros de imagen pic_parameter_set_rbsp ( )	no VCL
9	Delimitador de unidades de acceso access_unit_delimiter_rbsp ( )	no VCL
10-11	Reservado	n/a
12	Datos de relleno filler_data_rbsp ( )	no VCL
13	Extensión de conjunto de parámetros de secuencia seq_parameter_set_extension_rbsp ( )	no VCL
14	Conjunto de parámetros de adaptación aps_rbsp ( )	no VCL
15	Subconjunto del conjunto de parámetros de secuencia subset_seq_parameter_set_rbsp ( )	no VCL
16-19	Reservado	n/a
20 ó SVC_NAL	Extensión de franja codificada slice_layer_extension_rbsp ( )	VCL
21-23	Reservado	n/a
24..63	Sin especificar	no VCL

dependency_id representa un número de identificación (ID) que indica una relación de dependencia de cada unidad de NAL.

quality_id representa un número de ID transmitido para indicar el nivel de calidad de cada unidad de NAL.

Cuando el valor de **singleloop_decoding_flag** es 1, la compensación de movimiento se lleva a cabo solamente en la capa de más arriba de un flujo continuo de bits escalable. Cuando el valor de **singleloop_decoding_flag** es 0, la compensación de movimiento se permite en todas las capas.

slice_type especifica un tipo de codificación de una franja de acuerdo con la Tabla 48.

<Tabla 48>

slice_type	Nombre de slice_type
0, 5	EP (franja P en extensión escalable)
1, 6	EB (franja B en extensión escalable)
2, 7	EI (franja I en extensión escalable)

Cuando el valor de **inter_layer_intra_prediction_flag** es 1, especifica que la operación de predicción intercapa intra se usa de manera adaptativa en la unidad de codificación. De lo contrario, la predicción intercapa intra no se usa. Cuando **inter_layer_intra_prediction_flag** no está presente, el valor de inter_layer_intra_prediction_flag se puede deducir que es 0. ILIntraPredFlag se puede obtener de la manera siguiente.
- Cuando el valor de singleloop_decoding_flag es 1 y cuando el valor de inter_layer_intra_prediction_flag es 1 y slice_type es EI, el valor de ILIntraPredFlag se fija a 1. En caso contrario, el valor de ILIntraPredFlag se fija a 0.

- Cuando el valor de singleloop_decoding_flag es 0, el valor de ILIntraPredFlag se fija igual que el valor de inter_layer_intra_prediction_flag.

Cuando el valor de **inter_layer_syntax_prediction_flag** es 1, esto especifica que la operación de predicción de sintaxis intercapa se usa de manera adaptativa en la unidad de codificación. De lo contrario, no se usa la predicción de sintaxis intercapa. Cuando inter_layer_syntax_prediction_flag no está presente, el valor de inter_layer_syntax_prediction_flag se puede deducir que es 0. ILSyntaxPredFlag se puede obtener de la manera siguiente.

- Cuando el valor de inter_layer_syntax_prediction_flag es 1 y slice_type no es EI, el valor de ILSyntaxPredFlag se fija a 1. En caso contrario, el valor de ILSyntaxPredFlag se fija a 0.

Cuando el valor de **inter_layer_residual_prediction_flag** es 1, esto especifica que la operación de predicción residual intercapa se usa de manera adaptativa en la unidad de codificación. En caso contrario, no se usa la predicción residual intercapa. Cuando inter_layer_residual_prediction_flag no está presente, el valor de inter_layer_residual_prediction_flag se puede deducir que es 0. ILResPredFlag se puede obtener de la manera siguiente.

- Cuando el valor de inter_layer_residual_prediction_flag es 1 y slice_type no es EI, el valor de ILResPredFlag se fija a 1. En caso contrario, el valor de ILResPredFlag a 0.

cabac_init_idc especifica un índice para determinar una tabla de inicialización usada en un proceso de inicialización de una variable de contexto. cabac_init_idc puede tener un valor en un intervalo de 0 a 2.

il_mode es sintaxis que indica si utilizar información de una capa de base. il_mode tiene las siguientes funciones dependiendo del valor de singleloop_decoding_flag.

Cuando el valor de singleloop_decoding_flag es 1 y slice_type es EI y cuando el valor de il_mode es 1, la capa de base reconstruida se puede escalar basándose en una diferencia de resolución con respecto a la capa actual y, a continuación, se puede usar como información de predicción de una CU actual. Cuando el valor de il_mode es 0, la información de reconstrucción de la capa de base no se usa.

Cuando el valor de singleloop_decoding_flag es 1 y slice_type no es EI, y cuando el valor de il_mode es 1, la información de movimiento y la información de predicción intra de la capa de base se pueden escalar y copiar teniendo en cuenta la diferencia de resolución con respecto a la capa actual correspondiente para generar una señal de predicción, y su valor se puede usar como predictor. Cuando el valor de il_mode es 0, la información de reconstrucción de la capa de base no se usa.

Cuando el valor de singleloop_decoding_flag es 0 y el valor de il_mode es 1, la información de píxeles reconstruida de la capa de base se puede escalar teniendo en cuenta la resolución y, a continuación, se puede usar como información de predicción.

Cuando el valor de **il_res_mode** es 1, los datos residuales reconstruidos de la capa de base correspondiente se pueden escalar teniendo en cuenta la resolución y, a continuación, se pueden usar como predictor para el valor residual de la CU actual. Cuando el valor de il_res_mode es 0, los datos residuales reconstruidos de la capa de base correspondiente no se usan.

Cuando el valor de **inter_layer_differential_coding_flag** es 1, esto especifica que la operación de codificación diferencial intercapa se usa de manera adaptativa en la unidad de codificación. En caso contrario, no se usa la codificación diferencial intercapa. Cuando inter_layer_differential_coding_flag no está presente, se deduce que el valor de inter_layer_differential_coding_flag es 0. ILDiffCodingFlag se puede obtener de la manera siguiente.

Cuando el valor de singleloop_decoding_flag es 1, el valor de inter_layer_differential_coding_flag es 1, y slice_type es EI, el valor de ILDiffCodingFlag se fija a 1. En caso contrario, el valor de ILDiffCodingFlag se fija a 0.

- Cuando el valor de singleloop_decoding_flag no es 1, el valor de ILDiffCodingFlag se fija igual que el valor de inter_layer_differential_coding_flag.

Predicción de unidades intercapa

La señal de entrada usada para la codificación escalable puede ser diferente entre capas en cuanto a resolución, frecuencia de cuadro, profundidad de bits, formato de color, relación de aspecto y similares.

Al llevar a cabo la predicción intercapa teniendo en cuenta este punto, es posible reducir la redundancia y mejorar la eficiencia de codificación con respecto a la difusión simultánea (*simulcast*).

Por ejemplo, como método para reducir la cantidad de información redundante se puede usar un método de reducción de la cantidad de información sobre unidades de proceso, es decir, CU, PU, y TU, transmitida en una capa de mejora usando la información de una capa de base.

Al método de reducción de la información de unidades transmitida en la capa de mejora usando la información de unidades (CU, PU y/o TU) de la capa de base se le hace referencia como predicción de unidades intercapa.

La predicción de unidades intercapa la puede llevar a cabo el módulo de predicción o el módulo de predicción intercapa ilustrados en las FIGS. 1 a 4. En lo sucesivo en la presente, se supone, por motivos de comodidad en la explicación, que la predicción de unidades intercapa es llevada a cabo por el módulo de predicción.

Por ejemplo, cuando se realiza la predicción de unidades intercapa y el módulo de predicción adquiere información de unidades de una capa de base, se puede llevar a cabo una partición de la capa de mejora basándose en la información de unidades adquirida.

La FIG. 15 es un diagrama que ilustra esquemáticamente un ejemplo de predicción de información de unidades intercapa de acuerdo con la invención.

En el ejemplo ilustrado en la FIG. 15, se supone que la resolución de la capa de mejora es el doble de la resolución de la capa de base.

En la FIG. 15, la capa 0 puede ser una capa de base o una capa de referencia, y la capa 1 puede ser una capa de mejora o una capa actual.

En referencia a la FIG. 15, un bloque 1510 de LCU de la capa 0 se divide por particiones en diversas CUs, Pus o TUs. En este caso, a la información de partición de las CUs, Pus y/o TUs se le hace referencia como información de CU o información de unidades por motivos de comodidad en la explicación.

El módulo de predicción puede generar información 1520 de CU de referencia la cual es información obtenida por escalado en sentido ascendente de la información de CU de la capa de base. El módulo de predicción puede obtener la información de CU de la capa 0 usando la información 1520 de CU de referencia.

Por ejemplo, cuando la información de CU de la capa de base se obtiene a partir de la información 1520 de CU de referencia y se aplica a la capa 1, como estructura de LCU de la capa 1 se puede usar una estructura de unidad tal como la LCU0 1530. Cuando no se usa la información 1520 de CU de referencia, es decir, cuando no se usa la información de CU de la capa de base, como estructura de LCU de la capa 1 se puede usar una estructura de unidad tal como la LCU1 1540.

El uso de la información de CU (información de unidades) de la capa 0 puede resultar útil o no para mejorar la eficiencia de codificación. Por consiguiente, el codificador de vídeo puede señalizar de manera adaptativa si usar la información de CU de la capa 0.

Por motivos de comodidad en la explicación, se ha descrito que la información de unidades es información de partición de CU, PU y TU, pero la información de unidades puede incluir información sobre CU, PU y TU además de la estructura de partición de CU, PU y TU. En otras palabras, la información de CU de la capa de base puede ser una información de estructura en árbol o de partición, o puede ser información de PU (información sobre si usar un modo de omisión o un modo sin omisión, la dirección de la predicción, el vector de movimiento, índice de referencia y similares), o puede incluir ambas.

Por ejemplo, cuando la predicción de unidades intercapa está asociada a la predicción de la información de árbol/partición, se puede usar la siguiente sintaxis.

La Tabla 49 muestra un ejemplo de sintaxis de datos de franja para una capa de extensión escalable.

<Tabla 49>

slice_data( ) {	Descriptor
CurrTbAddr = first_tb_in_slice	
moreDataFlag = 1	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlagIdx = -1	
do {	
xCU = HorLumaLocation(CurrTbAddr)	
yCU = VerLumaLocation(CurrTbAddr)	
IsCuQpDeltaCoded = 0	
bl_tree_info_skip_flag	ae(v)
coding_tree(xCU, yCU, Log2TbSize)	
if(!entropy_coding_mode_flag)	
moreDataFlag = more_rbsp_data( )	
else {	
end_of_slice_flag	ae(v)
moreDataFlag = !end_of_slice_flag	
}	
CurrTbAddr = NextTbAddress(CurrTbAddr)	
} while(moreDataFlag)	
}	

La Tabla 50 muestra un ejemplo de sintaxis de árbol de codificación para una capa de extensión escalable.

<Tabla 50>

coding_tree(x0, y0, log2CUSize) {	Descriptor
if(x0 + (1 << log2CUSize) <= $PicWidthInSamples_L$ && y0 + (1 << log2CUSize) <= $PicHeightInSamples_L$ && log2CUSize > Log2MinCUSize)	
if(!bl_tree_info_skip_flag && BLSplitInfo[x0][y0])	
split_coding_unit_flag[x0 **][** y0 **]**	u(1) | ae(v)
if(adaptive_loop_filter_flag && alf_cu_control_flag) {	
cuDepth = Log2MaxCUSize – log2CUSize	
if(cuDepth <= alf_cu_control_max_depth)	
if(cuDepth == alf_cu_control_max_depth ||	
split_coding_unit_flag[x0][y0] == 0)	
AlfCuFlagIdx++	
}	
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
....	

Cuando el valor de **bl_tree_info_skip_flag** es 1, esto indica que la información de árbol de la capa de base (capa de referencia o capa 0) se usa sin ningún cambio. Cuando el valor de **bl_tree_info_skip_flag** es 0, esto indica que la información de árbol de la capa de base no se usa.

Para ayudar a entender bl_tree_info_skip_flag se puede hacer referencia a la FIG. 5. Cuando el valor de bl_tree_info_skip_flag es 1, la información de árbol de la capa de base se escala en sentido ascendente para adaptarse a la resolución de la capa de mejora (la capa actual o capa 1) mediante escalado en sentido ascendente de la información de CU. Por consiguiente, los valores de split_coding_unit_flag de la unidad de codificación más grande (LCU) actual se pueden obtener de manera que sean los mismos valores que la información de partición escalada en sentido ascendente de la capa de base.

BLSplitInfo [x0] [y0] tiene un valor de 1 cuando hay presencia de información de partición en la información de CU escalada en sentido ascendente, y tiene un valor de 0 cuando la información de partición no está presente en la información de CU escalada en sentido ascendente.

Por ejemplo, cuando la resolución de la capa de mejora es el doble de la resolución de la capa de base y la capa de mejora y la capa de base tienen la misma profundidad de CU, la información de división escalada en sentido ascendente de la capa de base tiene una profundidad menor, en un paso, que la correspondiente de la información de partición de la capa de mejora.

Cuando la información de partición no está presente en la capa de base, la información de partición de división adicional con respecto a la información de partición cuando la información de división se predice a partir de la capa de base

escalada en sentido ascendente se puede señalizar fijando BLSplitInfo a 0 y transmitiendo información adicional (por ejemplo, split_coding_unit_flag) únicamente para la capa de mejora.

Se puede aplicar un método similar a TUs. Por ejemplo, la información de partición de TU para la capa actual se puede procesar usando información predeterminada de una bandera.

La Tabla 51 muestra un ejemplo de sintaxis de una unidad de codificación en extensión escalable de acuerdo con la invención. En el ejemplo mostrado en la Tabla 51, la transmisión de la información de partición de TU para la capa actual se puede omitir usando una bandera bl_tu_info_skip_flag.

<Tabla 51>

coding_unit(x0, y0, log2CUSize) {	**Descriptor**
if(entropy_coding_mode_flag && slice_type != I)	
skip_flag[x0][y0]	u(1) | ae(v)
...	
prediction_unit(x1, y1, log2CUSize – 1, log2CUSize – 1, 3)	
}	
}	
if(!pcm_flag) {	
bl_tu_info_skip_flag	u(1) | ae(v)
transform_tree(x0, y0, log2CUSize, 0, 0)	
transform_coeff(x0, y0, log2CUSize, 0, 0)	
transform_coeff(x0, y0, log2CUSize, 0, 1)	
transform_coeff(x0, y0, log2CUSize, 0, 2)	
}	
}	

La Tabla 52 muestra un ejemplo de sintaxis de árbol de transformada en extensión escalable de acuerdo con la invención.

<Tabla 52>

transform_tree(x0, y0, log2TrafoSize, trafoDepth, blkIdx) {	**Descriptor**
if(entropy_coding_mode_flag && trafoDepth == 0 && IntraSplitFlag == 0) {	
if(PredMode != MODE_INTRA)	
no_residual_data_flag	u(1) | ae(v)
residualDataPresentFlag = !no_residual_data_flag	
} else {	
.....	
if(log2TrafoSize <= Log2MaxTrafoSize && !intraSplitFlag && log2TrafoSize > Log2MinTrafoSize && trafoDepth < maxDepth && bl_tu_info_skip_flag)	
split_transform_flag[x0][y0][trafoDepth]	u(1)
} else	
cbp_and_split_transform	vlc(n,v)
}	
if(log2TrafoSize <= Log2MaxTrafoSize && log2TrafoSize > Log2MinTrafoSize && trafoDepth < maxDepth && !intraSplitFlag && entropy_coding_mode_flag && bl_tu_info_skip_flag)	
split_transform_flag[x0][y0][trafoDepth]	u(1) | ae(v)
if(PredMode != MODE_INTRA && log2TrafoSize <= Log2MaxTrafoSize && entropy_coding_mode_flag) {	
firstChromaCbf = (log2TrafoSize == Log2MaxTrafoSize || trafoDepth == 0 ? 1 : 0)	
if(firstChromaCbf || log2TrafoSize > Log2MinTrafoSize) {	
xBase = x0 – (x0 & ((1 << log2TrafoSize) – 1))	
....	
}	
}	

Tal como se ha descrito anteriormente, en las Tablas 51 y 52, cuando el valor de bl_tu_info_skip_flag es 1, la información de partición de TU escalada en sentido ascendente de la capa de base se puede usar en la capa actual sin ningún cambio. Cuando el valor de bl_tu_info_skip_flag es 0, la información de partición de TU para la capa actual se puede transmitir de manera independiente desde el codificador de vídeo.

Por otro lado, informaciones de CU/PU/TU se pueden combinar para su uso en la aplicación de la predicción de unidades intercapa. Con el fin de entender la invención, a continuación se describirá un ejemplo en el que la relación espacial de la capa de base y la capa de mejora es 2, es decir, un ejemplo en el que la diferencia de resolución entre dos capas es de dos veces.

La FIG. 16 es un diagrama que ilustra esquemáticamente un ejemplo en el que la predicción de unidades intercapa se aplica de acuerdo con la invención.

A continuación se describirá, en referencia a la FIG. 16, un método de remuestreo de una estructura de capa de base en el momento de la aplicación de la predicción de unidades intercapa.

La FIG. 16(a) ilustra una estructura de CU y PU de una capa de base.

En la FIG. 16(a), se supone que el bloque 1600 de capa base es un bloque intracodificado, la región 1620 de división es una partición de NxN, y la otra región es una partición de 2Nx2N.

En la FIG. 16(a), cuando se realiza un sobremuestreo doble sobre información de CU/PU de un bloque 1610 de referencia para su uso en una capa de mejora, en función del nivel (por ejemplo, nivel de CU o nivel de PU) de la estructura de partición de la capa de base a utilizar se puede obtener una estructura de partición como el bloque 1630 ilustrado en la FIG. 16(b) o una estructura de partición como el bloque 1640 ilustrado en la FIG. 16(c).

En este momento, cuando para la capa de mejora se usa solamente la información de CU de la capa de base, la estructura de partición de CU de la capa de mejora se puede construir como la estructura de partición del bloque 1630. En este caso, no se puede cubrir el caso en el que un vídeo de la capa de mejora tenga una estructura de partición de división adicional con respecto al bloque 1630.

Por el contrario, cuando la información de PU, además de la información de CU de la capa de base, se refleja en la estructura de partición de CU de la capa de mejora, la región de la capa de mejora correspondiente a una parte que tiene una partición de división adicional (por ejemplo, partición de NxN) diferente a la partición de 2Nx2N de la capa base como el bloque 1640 ilustrado en la FIG. 16(c) se puede dividir adicionalmente en particiones en CUs. Por consiguiente, el vídeo de la capa de mejora puede presentar una estructura de partición con división adicional.

Por otro lado, la FIG. 16 ilustra un ejemplo en el que la capa de base es un bloque intracodificado, pero puede aplicarse el mismo método a un caso en el que la capa de base sea un bloque intercodificado.

La FIG. 17 es un diagrama que ilustra esquemáticamente otro ejemplo en el que se aplica la predicción de unidades intercapa de acuerdo con la invención.

En la FIG. 17, se supone que el bloque 1700 de una capa de base es un bloque intercodificado, la región 1720 es una partición de Nx2N, y la región 1730 es una partición de 2NxN.

En el caso del bloque intercodificado, además de la partición de 2Nx2N y la partición de NxN se pueden usar diversos tipos de partición, tales como una partición de 2NxN, una partición de Nx2N, una partición de 2NxnU, una partición de 2NxnD, una partición de nLx2N y una partición de nRx2N.

Cuando se realiza un sobremuestreo doble sobre información de CU/PU de un bloque 1710 de referencia de una capa de base para su uso en una capa de mejora, en función del nivel (por ejemplo, nivel de CU o nivel de PU) de la estructura de partición de la capa de base a utilizar se puede obtener una estructura de partición como el bloque 1740 ilustrado en la FIG. 16(b) o una estructura de partición como el bloque 1750 ilustrado en la FIG. 17(c).

Cuando, para la capa de mejora, se usa solamente la información de CU de la capa de base, la estructura de partición de CU de la capa de mejora se puede construir como la estructura de partición del bloque 1740 ilustrado en la FIG. 17(b) sobremuestreando solamente la estructura de partición de CU excepto para las estructuras de la región 1720 y la región 1730 en el bloque 1710 de referencia.

Cuando la información de PU de la capa de base se refleja en la estructura de partición de CU de la capa de mejora, la región de la capa de mejora correspondiente a una parte que presenta una partición (por ejemplo, la partición de la región 1720 ó la región 1730) diferente a la partición de 2Nx2N en la capa de base se puede dividir adicionalmente por particiones en CUs tal como se ilustra en la FIG. 17(c).

En otras palabras, cuando la información de PU de la capa de base se refleja en la estructura de partición de CU de la capa de mejora en las FIGS. 16 y 17, la región de la capa de mejora correspondiente a la región que incluye la partición de PU en la capa de base se puede dividir por particiones en CUs.

Por otro lado, a la capa de mejora se le puede aplicar un método de uso selectivo de la estructura de partición de la capa de base.

La FIG. 18 es un diagrama que ilustra esquemáticamente todavía otro ejemplo en el que la predicción de unidades intercapa se aplica de acuerdo con la invención.

Se describirá, en referencia a la FIG. 18, un método de reutilización de la estructura de partición de la capa de base en la capa de mejora.

Tal como se ilustra en la FIG. 18(a), se supone que un bloque 1800 de una capa de base presenta particiones cuadradas, tales como 2Nx2N ó NxN, y particiones rectangulares, tales como 2NxnU, 2NxnD, nLx2N, ó nRx2N.

En este caso, puesto que una CU se divide en forma de solamente un cuadrado (2Nx2N ó NxN), puede considerarse el método de uso selectivo de la estructura de partición de la capa de base en la capa de mejora.

Por ejemplo, tal como se ilustra en la FIG. 18(b), la estructura de partición de un bloque 1810 de referencia se puede reflejar en forma de estructura de partición de un bloque 1840 de la capa de mejora.

Tal como se ilustra en la FIG. 18(c), las particiones cuadradas 1820 en la estructura de partición de PU de un bloque de referencia se pueden reflejar en la estructura de partición de CU de la capa de mejora, y las particiones rectangulares 1830 en la estructura de partición de PU del bloque de referencia no se pueden reflejar en la estructura de partición de CU de la capa mejora.

De esta manera, el método de combinación y uso de informaciones de CU/PU/TU puede ser usado selectivamente por unidades arbitrarias. Es decir, se puede determinar si seleccionar el método de combinación y uso de las informaciones de CU/PU/TU de la capa de base para su utilización en la capa de mejora, sobre la base de unidades arbitrarias tales como una secuencia, un grupo de imágenes, una imagen individual, diversas franjas, una única franja, diversas LCUs y una LCU de señalización.

Predicción de textura intercapa adaptativa

Es posible mejorar la calidad de una imagen de referencia aplicando un filtro a la predicción intercapa. Por ejemplo, cuando se lleva a cabo la predicción intercapa intra, se puede aplicar un filtro a una imagen de referencia para mejorar la calidad de la imagen de referencia.

En el caso de la escalabilidad espacial, se puede aplicar un filtro de sobremuestreo a información de textura de una capa inferior para ajustar la resolución entre capas de manera que sea la misma y, a continuación, la información de textura ajustada de la capa inferior se puede usar como imagen de referencia.

En este momento, la información de textura de la capa inferior a la que se ha aplicado el filtro de sobremuestreo se puede someter a un filtrado adicional y, a continuación, se puede usar como imagen de referencia para predecir una capa superior. En esta descripción, a este método de predicción se le hace referencia como predicción de textura intercapa adaptativa.

La FIG. 19 es un diagrama que ilustra esquemáticamente un ejemplo de un método de ejecución de la predicción de textura intercapa adaptativa de acuerdo con la invención.

En referencia a la FIG. 19, el módulo 1910 de predicción de textura se corresponde con el módulo de predicción de textura de la FIG. 1. Por consiguiente, el módulo de predicción de textura puede llevar a cabo un reescalado junto con la predicción de textura, si ello fuera necesario.

El filtrado aplicado en la capa de base por un módulo 1950 de filtrado sirve para reducir la diferencia con respecto a una secuencia de entrada convertida en sentido descendente por un conversor 1940 de sentido descendente.

El filtrado de la textura, que se ha sobremuestreado por medio del módulo 1930 de sobremuestreo, en la capa de mejora por parte del módulo 1920 de filtrado, sirve para reducir el error entre la textura sobremuestreada de la capa de base y la secuencia de entrada.

El filtro aplicado únicamente tiene que ser un filtro con capacidad de lograr los servicios antes mencionados. Por ejemplo, el filtro aplicado por el módulo 1920 de filtrado puede ser un filtro especificado en un tamaño de tomas predeterminada y coeficientes predeterminados de antemano por el codificador/descodificador de vídeo, o puede ser un filtro cuyos parámetros de filtro (tales como el tamaño de las tomas y los coeficientes) se señalizan de forma adaptativa.

Cuando se usa la predicción de textura intercapa adaptativa, es posible mejorar la eficiencia de codificación debido a la mejora en la calidad de la imagen de referencia.

Cuando se usa la predicción de textura intercapa adaptativa, se mejora la calidad de la imagen de referencia, pero aumenta la complejidad y es necesario codificar/descodificar adicionalmente los parámetros del filtro. Por consiguiente, el hecho de usar o no la predicción de textura intercapa adaptativa se puede determinar en función de las situaciones (por ejemplo, rendimiento del codificador de vídeo y del descodificador de vídeo y un error entre una secuencia de entrada y una textura sobremuestreada de una capa de base).

Por lo tanto, el filtro usado por el módulo 1920 de filtrado puede ser un filtro usado de forma adaptativa en un bucle de codificación/descodificación. Por motivos de comodidad en la explicación, en esta descripción se supone que el filtro usado por el módulo 1920 es un ALF intercapa. Cuando el codificador/descodificador de vídeo usa el ALF como filtro dentro del bucle, en calidad del ALF intercapa se puede usar un filtro de bucle adaptativo (ALF) como filtro dentro del bucle.

Para usar la predicción de textura intercapa adaptativa se requiere una bandera que indique si utilizar el ALF intercapa. Se considera que la bandera que indica si utilizar el ALF intercapa es inter_layer_adaptive_loop_filter_enabled_flag.

Cuando el valor de inter_layer_adaptive_loop_filter_enabled_flag es 0, esto indica que no se usa el ALF intercapa. Cuando el valor de inter_layer_adaptive_loop_filter_enabled_flag es 1, esto indica que se usa el ALF intercapa.

La bandera inter_layer_adaptive_loop_filter_enabled_flag se puede definir en un conjunto de parámetros de secuencia.

La Tabla 53 muestra un ejemplo del conjunto de parámetros de secuencia para aplicar la predicción de textura intercapa adaptativa de acuerdo con la invención.

<Tabla 53>

seq_parameter_set_rbsp( ) {	**Descriptor**
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
...	
adaptive_loop_filter_enabled_flag	u(1)
Inter_layer_adaptive_loop_filter_enabled_flag	u(1)
pcm_loop_filter_disable_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
temporal_id_nesting_flag	u(1)
rbsp_trailing_bits( )	
}	

La bandera inter_layer_adaptive_loop_filter_enabled_flag se puede definir en un encabezamiento de franja diferente al conjunto de parámetros de secuencia.

La Tabla 54 muestra un ejemplo del encabezamiento de franja para aplicar la predicción de textura intercapa adaptativa de acuerdo con la invención.

<Tabla 54>

slice_header( ) {	**Descriptor**
...	
Inter_layer_adaptive_loop_filter_enabled_flag	**u(1)**
if(adaptive_loop_filter_enabled_flag) {	
if(!shared_pps_info_enabled_flag)	
alf_param( )	
alf_cu_control_param( )	
}	
if(inter_layer_adaptive_loop_filter_enabled_flag){	
alf_param()	
alf_cu_control_param( )	
}	
...	

Cuando el codificador de vídeo y el descodificador de vídeo usan el ALF como filtro dentro del bucle, se puede aplicar de manera similar como ALF intercapa la sintaxis usada para utilizar el ALF en un conjunto de parámetros de imagen.

La Tabla 55 muestra un ejemplo del conjunto de parámetros de imagen para aplicar la predicción de textura intercapa adaptativa de acuerdo con la invención.

<Tabla 55>

pic_parameter_set_rbsp( ) {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag	u(1)
shared_pps_info_enabled_flag	u(1)
if(shared_pps_info_enabled_flag){	
if(adaptive_loop_filter_enabled_flag)	
alf_param( )	
if(Inter_layer_adaptive_loop_filter_enabled_flag)	
alf_param( )	
}	
if(cu_qp_delta_enabled_flag)	
max_cu_qp_delta_depth	u(4)
rbsp_trailing_bits( )	
}	

En la Tabla 55, cuando el valor de shared_pps_info_enabled_flag es 1, se puede usar el conjunto de parámetros del ALF intercapa. A diferencia de la Tabla 55, se puede usar un método de utilización de una bandera para aplicar el ALF intercapa independientemente del ALF como filtro dentro del bucle.

La Tabla 56 muestra un ejemplo de sintaxis de datos de franja cuando shared_pps_info_enabled_flag se aplica al ALF intercapa de acuerdo con la invención.

<Tabla 56>

slice_header( ) {	Descriptor
...	
if(adaptive_loop_filter_enabled_flag) {	
if(!shared_pps_info_enabled_flag)	
alf_param( )	
alf_cu_control_param( )	
}	
if(inter_layer_adaptive_loop_filter_enabled_flag){	
if(!shared_pps_info_enabled_flag)	
alf_param()	
alf_cu_control_param( )	
}	
...	

Predicción de parámetros de filtro intercapa

En calidad de filtro utilizable en un proceso de filtrado de bucle se pueden usar tres filtros de entre un filtro antibloques, una compensación adaptativa según muestras (SAO), y un filtro de bucle adaptativo (ALF).

Como filtro de bucle se pueden usar tres filtros o se puede usar solamente una parte de los mismos. Por ejemplo, se puede usar solamente el filtro antibloques o se pueden usar solamente el filtro antibloques y la SAO.

El codificador de vídeo puede determinar los parámetros del filtro de manera que consigan que una imagen reconstruida se aproxime al máximo a una imagen de entrada y puede transmitir al descodificador de vídeo los parámetros de filtro determinados. Por consiguiente, puesto que las imágenes en la capa de base y la capa de mejora son muy similares entre sí en cuanto a características de la codificación escalable, existe una alta posibilidad de que los parámetros de filtro en dos matrices sean similares entre sí.

Por lo tanto, en la capa de mejora se puede considerar un método de reutilización de los parámetros, que se han usado en la capa de base. A esto se le hace referencia como predicción de parámetros de filtro intercapa.

La predicción de parámetros de filtro intercapa la puede llevar a cabo el módulo de predicción o el módulo de predicción intercapa ilustrados en las FIGS. 1 a 4, o la puede llevar a cabo el módulo de filtrado para la capa de mejora.

La FIG. 20 es un diagrama que ilustra esquemáticamente la predicción de parámetros de filtro intercapa de acuerdo con la invención.

En referencia a la FIG. 20, los parámetros de filtro usados en un módulo 2020 de filtrado dentro del bucle para la capa de base se pueden reutilizar en un módulo 2010 de filtrado dentro del bucle para una capa actual.

Los parámetros de SAO se pueden transmitir mediante el uso de un conjunto de parámetros adaptativo. Los parámetros de ALF también se pueden transmitir mediante el uso del conjunto de parámetros adaptativo.

La Tabla 57 muestra un ejemplo del conjunto de parámetros adaptativo en extensión escalable de acuerdo con la invención.

<Tabla 57>

aps_rbsp_in_scalable_extension ( ) {	**Descriptor**
aps_id	ue(v)
aps_sample_adaptive_offset_flag	u(1)
aps_adaptive_loop_filter_flag	u(1)
if(aps_sample_adaptive_offset_flag || aps_adaptive_loop_filter_flag) {	
aps_cabac_use_flag	u(1)
if(aps_cabac_use_flag) {	
aps_cabac_init_idc	ue(v)
aps_cabac_init_qp_minus26	se(v)
}	
}	
/* Insert non-CABAC stuff above this line */	
base_pred_alf_param_flag	u(1)
if(aps_adaptive_loop_filter_flag && ! base_pred_alf_param_flag) {	
alf_data_byte_count /* to enable skipping past data without parsing it */	u(8)
/* byte_align() this byte align to happen between the non-CABAC and CABAC parts of the alf_param() Once there is an all CABAC alf_param(), enable this byte_align() */	
alf_param( )	
byte_align()	
}	
/* insert CABAC stuff below this line; make sure its byte-aligned */	
base_pred_sao_param_flag	u(1)
if(aps_sample_adaptive_offset_flag && ! base_pred_sao_param_flag) {	
sao_data_byte_count /* to enable skipping past data without parsing it */	u(8)
byte_align ()	
sao_param( )	
/* byte_align() this final byte align unnecessary as being taken care of by rbsp_trailing_bits() */	
rbsp_trailing_bits( )	
}	

En este caso, **aps_id** es un identificador destinado a identificar el conjunto de parámetros adaptativo al que se hace referencia en un encabezamiento de franja. El valor de aps_id está en un intervalo de 0 a TBD y depende del nivel/perfilado.

Cuando el valor de **aps_sample_adaptive_offset_flag** es 1, esto indica que la SAO está en ON para una franja a la que se hace referencia en ese momento en el conjunto de parámetros adaptativo. Cuando el valor de aps_sample_adaptive_offset_flag es 0, esto indica que la SAO está en OFF para una franja a la que se hace referencia en ese momento en el conjunto de parámetros adaptativo. Cuando el conjunto de parámetros adaptativo activado no está presente, se deduce que el valor de aps_sample_adaptive_offset_flag es 0.

Cuando el valor de **aps_adaptive_loop_filter_flag** es 1, esto indica que el ALF está en ON para una franja a la que se hace referencia en ese momento en el conjunto de parámetros adaptativo. Cuando el valor de aps_adaptive_loop_filter_flag es 0, esto indica que el ALF está en OFF para una franja a la que se hace referencia en ese momento en el conjunto de parámetros adaptativo. Cuando no está presente el conjunto de parámetros adaptativo activado, se deduce que el valor de aps_adaptive_loop_filter_flag es 0.

El proceso de descodificación CABAC se aplica a sao_param( ) y alf_param( ) cuando el valor de **aps_cabac_use_flag** es 1, y el proceso de descodificación CAVLC se aplica a sao_param( ) y alf_param( ) cuando el valor de aps_cabac_use_flag es 0.

aps_cabac_init_idc especifica un índice para determinar una tabla de inicialización usada en el proceso de inicialización para variables de contexto de la SAO y el ALF. El valor de aps_cabac_init_idc está en el intervalo de 0 a 2.

aps_cabac_init_qp_minus26 especifica el valor del parámetro de cuantificación-26. En este caso, el parámetro de cuantificación se usa para el proceso de inicialización para las variables de contexto de la SAO y el ALF.

alf_data_byte_count y **sao_data_byte_point** especifica el número de bytes.

Cuando el valor de **base_pred_alf_param_flag** es 1, esto indica que el parámetro de ALF utilizado en la capa de base se usa en una capa actual. Cuando el valor de base_pred_alf_param_flag es 0, esto indica que el parámetro de ALF para la capa actual se usa en la capa actual.

Cuando el valor de **base_pred_sao_param_flag** es 1, esto indica que el parámetro de SAO usado en la capa de base se utiliza en una capa actual. Cuando el valor de base_pred_sao_param_flag es 0, esto indica que el parámetro de SAO para la capa actual no se usa en la capa actual.

Codificación diferencial intercapa de imágenes

Se supone que la imagen reconstruida de una capa de base es R_{BL} y la imagen obtenida sobremuestreando R_{BL} en función de la resolución de una capa de mejora es UR_{BL}. Se supone también que la imagen reconstruida de la capa de mejora es R_{EL}.

La imagen reconstruida puede ser una imagen antes de aplicar a la misma un filtrado dentro del bucle. La imagen reconstruida puede ser una imagen después de aplicar una parte de los filtros dentro del bucle (el filtro antibloques, el filtro de compensación adaptativa según muestras y/o el filtro de bucle adaptativo) a la misma. Adicionalmente, la imagen reconstruida puede ser una imagen después de aplicar a la misma todos los filtros dentro del bucle.

En este caso, cuando una imagen diferencial obtenida al restar el valor de UR_{BL} con respecto al valor de R_{BL} se define como D, se puede realizar una codificación/descodificación independiente en el dominio de las imágenes D. En esta descripción, a este método se le hace referencia como codificación diferencial intercapa de imágenes o modo diferencial intercapa (modo IL-Diff).

El modo diferencial intercapa se puede aplicar a las unidades de secuencia, imagen, franja, CU más grande (LCU), unidad de codificación (CU), o unidad de predicción (PU). En la unidad de proceso a la cual se aplica el modo diferencial intercapa, se puede transmitir, desde el codificador de vídeo al descodificador de vídeo, una bandera que indica si utilizar el modo diferencial intercapa.

Sobre la unidad de proceso en la que se aplica el modo diferencial intercapa no se lleva a cabo una codificación (codificación/descodificación) usando diferentes escalabilidades, sino que solamente puede usarse una codificación (codificación/descodificación) de una sola capa. En este caso, es posible guardar bits para indicar si se lleva a cabo una codificación usando diferentes escalabilidades.

El modo diferencial intercapa puede ser llevado a cabo por el módulo de predicción o el módulo de predicción intercapa ilustrados en las FIGS. 1 a 4. Por motivos de comodidad en la explicación, se supone que el modo diferencial intercapa es llevado a cabo por el módulo de predicción.

(1) Predicción intra para el modo IL-Diff

La FIG. 21 es un diagrama que ilustra esquemáticamente un método de ejecución de predicción intra cuando se usa el modo diferencial intercapa de acuerdo con la invención.

En referencia a la FIG. 21, una imagen 2100 de una capa de mejora incluye una región reconstruida 2105 y una región no reconstruida 2110 antes y después de un bloque actual 2115. Se puede obtener una imagen reconstruida R_{EL} a partir de la región reconstruida 2105. Cuando se reconstruye la imagen 2100 de la capa de mejora, la imagen 2100 puede ser la imagen reconstruida R_{EL}.

Por otro lado, una imagen UR_{BL} 2125 obtenida mediante el sobremuestreo de una imagen reconstruida R_{BL} 2120 de una capa de base incluye un bloque P_{BL} 2130 correspondiente al bloque actual 2115.

En el proceso de codificación, el módulo de predicción (el módulo de predicción del codificador de vídeo) puede obtener la diferencia D entre la imagen reconstruida de la capa de base y la imagen reconstruida de la capa de mejora según expone la Expresión 4.

<Expresión 4>

$$D = R_{EL} - UR_{BL}$$

En la Expresión 4, como R_{EL} puede usarse una imagen en la que no se aplica un filtro dentro de bucle, tal como el filtro antibloques, la SAO, o el ALF, debido a la presencia de la región no reconstruida 2110.

Puesto que se reconstruyen todas las regiones de la imagen 2120 de la capa de base, R_{BL} puede ser una imagen reconstruida en la cual se aplica la totalidad o una parte de los filtros dentro de bucle o puede ser una imagen reconstruida en la cual no se aplican los filtros dentro de bucle.

El módulo de predicción puede llevar a cabo la predicción intra sobre un bloque actual 2155 en referencia a valores de píxel de la región reconstruida 2145 excepto para la región no reconstruida 2150 en la imagen diferencia D 2140.

En el proceso de descodificación, el módulo de predicción (el módulo de predicción del descodificador de vídeo) puede reconstruir un bloque actual usando valores de bloque P_{BL} de UR_{BL} presentes en la misma posición que el bloque actual según expone la Expresión 5.

<Expresión 5>

$$R_{EL} = P_D + P_{BL} + RES$$

En la Expresión 5, P_D representa un bloque predicho construido llevando a cabo la predicción intra sobre la región reconstruida de la imagen diferencial D y RES representa un bloque residual.

(2) Predicción inter para el modo IL-Diff

Para llevar a cabo la predicción inter sobre un bloque actual en el momento de la aplicación del modo diferencial intercapa, el módulo de predicción genera una imagen diferencial D_R para una imagen de referencia de la imagen actual. Por ejemplo, el módulo de predicción genera la imagen diferencial D_R para la imagen de referencia de la imagen actual usando la imagen reconstruida correspondiente a una imagen de capa de mejora de la imagen de referencia y la imagen reconstruida correspondiente a una imagen de capa de base de la imagen de referencia.

El módulo de predicción puede generar un bloque predicho P_D en el dominio de las imágenes diferenciales del bloque actual sobre la base de la imagen diferencial D_R de la imagen de referencia.

El módulo de predicción puede reconstruir el bloque actual usando el bloque predicho según expone la Expresión 6.

<Expresión 6>

$$R_{EL} = P_D + P_{BL} + RES$$

En la Expresión 6, R_{EL} representa un bloque actual reconstruido en la capa de mejora. P_{BL} representa un bloque presente en la misma posición que el bloque actual en UR_{BL}, y RES representa un bloque residual.

La imagen diferencial D_R de la imagen de referencia se puede generar de antemano y se puede almacenar en una memoria intermedia de imágenes descodificadas (DPB). La DPB se puede corresponder con la memoria descrita en referencia a las FIGS. 1 a 3.

La imagen diferencial D_R de la imagen de referencia se puede calcular para un bloque especificado en una posición requerida para reconstruir el bloque actual con la información de movimiento del bloque actual cada vez que se genera R_{EL}.

En la predicción inter en el modo diferencial intercapa, a diferencia de la predicción intra en el modo diferencial intercapa, como imagen de referencia reconstruida en la capa de mejora en el momento de generar la imagen diferencial de la imagen de referencia se puede usar una imagen reconstruida en la cual se aplica una parte o la totalidad de los filtros dentro de bucle así como una imagen reconstruida en la cual no se aplican los filtros dentro de bucle.

En esta descripción, por motivos de comodidad en la explicación, a una matriz de muestras reconstruida en un instante de tiempo específico (por ejemplo, recuento de orden de imagen (POC) o unidad de acceso (AU)) por capas en una estructura multicapa que soporta la codificación de vídeo escalable se le hace referencia como “imagen”.

En relación con esto, una matriz de muestras reconstruida o que se va a reconstruir en un instante de tiempo específico en una capa (capa actual) descodificada y a la que se ha dado salida, se puede definir como una imagen de manera que se diferencie con respecto a una matriz de muestras reconstruida o que se va a reconstruir en una capa de referencia. La matriz de muestras reconstruida o que se va a reconstruir en la capa de referencia se puede denominar representación, imagen de capa de referencia, matriz de muestras de capa de referencia, textura de capa de referencia, y similares. En este caso, desde cada AU se puede dar salida a una imagen descodificada (codificada) reconstruida en la capa actual.

Aunque los métodos en el sistema ejemplificativo antes mencionado se han descrito sobre la base de diagramas de flujo que incluyen una serie de etapas o bloques, la invención no se limita al orden de las etapas y una cierta etapa se puede llevar a cabo en una etapa o un orden diferente al descrito anteriormente o al mismo tiempo que el descrito anteriormente. Las realizaciones antes mencionadas pueden incluir diversos ejemplos. Por lo tanto, la invención incluye toda sustitución, corrección y modificación que encaje con las reivindicaciones adjuntas.

REIVINDICACIONES

1. Método de predicción intercapa que comprende:

recibir información que indica un modo de predicción de un bloque actual (900), en donde el modo de predicción es un modo de fusión;

recibir información de índices que indica un candidato de una lista de candidatos de fusión;

obtener (S1010) información de movimiento intercapa de una capa (1100) de referencia; y

construir la lista de candidatos de fusión, en donde la lista de candidatos de fusión incluye candidatos espaciales que representan información de movimiento de bloques vecinos (A_0, A_1, B_0, B_1, B_2) del bloque actual (900) y un candidato intercapa que representa la información de movimiento intercapa; y

cuando el candidato indicado por la información de índices es el candidato intercapa, predecir (S1020) el bloque actual (900) en una capa actual (1110) usando la información de movimiento intercapa representada por el candidato intercapa seleccionado de la lista de candidatos de fusión sobre la base de la información de índices,

en donde la información de movimiento intercapa incluye un vector de movimiento intercapa obtenido a partir de la capa (1100) de referencia, y

en donde el vector de movimiento intercapa se obtiene escalando un vector de movimiento de la capa (1100) de referencia sobre la base de una relación de resolución de la capa (1100) de referencia y la capa actual (1110).

2. Método de predicción intercapa según la reivindicación 1, en el que el vector de movimiento de la capa (1100) de referencia es un vector de movimiento en una posición de referencia (LT, RT, C0, C1, C2, C3, LB, RB), que se corresponde con una posición para especificar el bloque actual (900) en la capa (1100) de referencia.

3. Método de predicción intercapa según la reivindicación 2, en el que la posición para especificar el bloque actual (900) es una posición superior izquierda (LT) del bloque actual.

4. Método de predicción intercapa según la reivindicación 2, en el que la posición de referencia (LT, RT, C0, C1, C2, C3, LB, RB) se obtiene escalando la posición destinada a especificar el bloque actual sobre la base de la relación de resolución de la capa (1100) de referencia y la capa actual (1110).

5. Método de predicción intercapa según la reivindicación 2, en el que el vector de movimiento de la posición de referencia (LT, RT, C0, C1, C2, C3, LB, RB) es un vector de movimiento de un bloque de predicción que incluye la posición de referencia.

6. Método de predicción intercapa según la reivindicación 1,

en el que el candidato intercapa se añade al principio de la lista de candidatos de fusión o al final de la lista de candidatos de fusión.

7. Método de predicción intercapa según la reivindicación 1, en el que el candidato intercapa se añade a la lista de candidatos de fusión después de candidatos temporales de la capa actual o después de candidatos espaciales de la capa actual.

8. Descodificador de vídeo escalable que comprende:

un módulo (315, 355) de descodificación entrópica que recibe información que indica un modo de predicción de un bloque actual (900), e información de índices recibida que indica un candidato de una lista de candidatos de fusión, en donde el modo de predicción es un modo de fusión;

un primer módulo (250, 450) de predicción que predice una capa (1100) de referencia; y

un segundo módulo (210, 410) de predicción que obtiene información de movimiento intercapa de la capa de referencia, construye la lista de candidatos de fusión, predice el bloque actual (900) en una capa actual (1110) usando la información de movimiento intercapa representada por un candidato intercapa seleccionado de la lista de candidatos de fusión sobre la base de la información de índices,

en donde la lista de candidatos de fusión incluye candidatos que representan información de movimiento de un bloque vecino (A_0, A_1, B_0, B_1, B_2) del bloque actual (900) y el candidato intercapa que representa información de movimiento intercapa,

en donde el candidato indicado por la información de índices es el candidato intercapa,

en donde la información de movimiento intercapa incluye un vector de movimiento intercapa obtenido a partir de la capa (1100) de referencia, y

en donde el vector de movimiento intercapa se obtiene escalando un vector de movimiento de la capa (1100) de referencia sobre la base de una relación de resolución entre la capa (1100) de referencia y la capa actual (1110).

FIG. 1

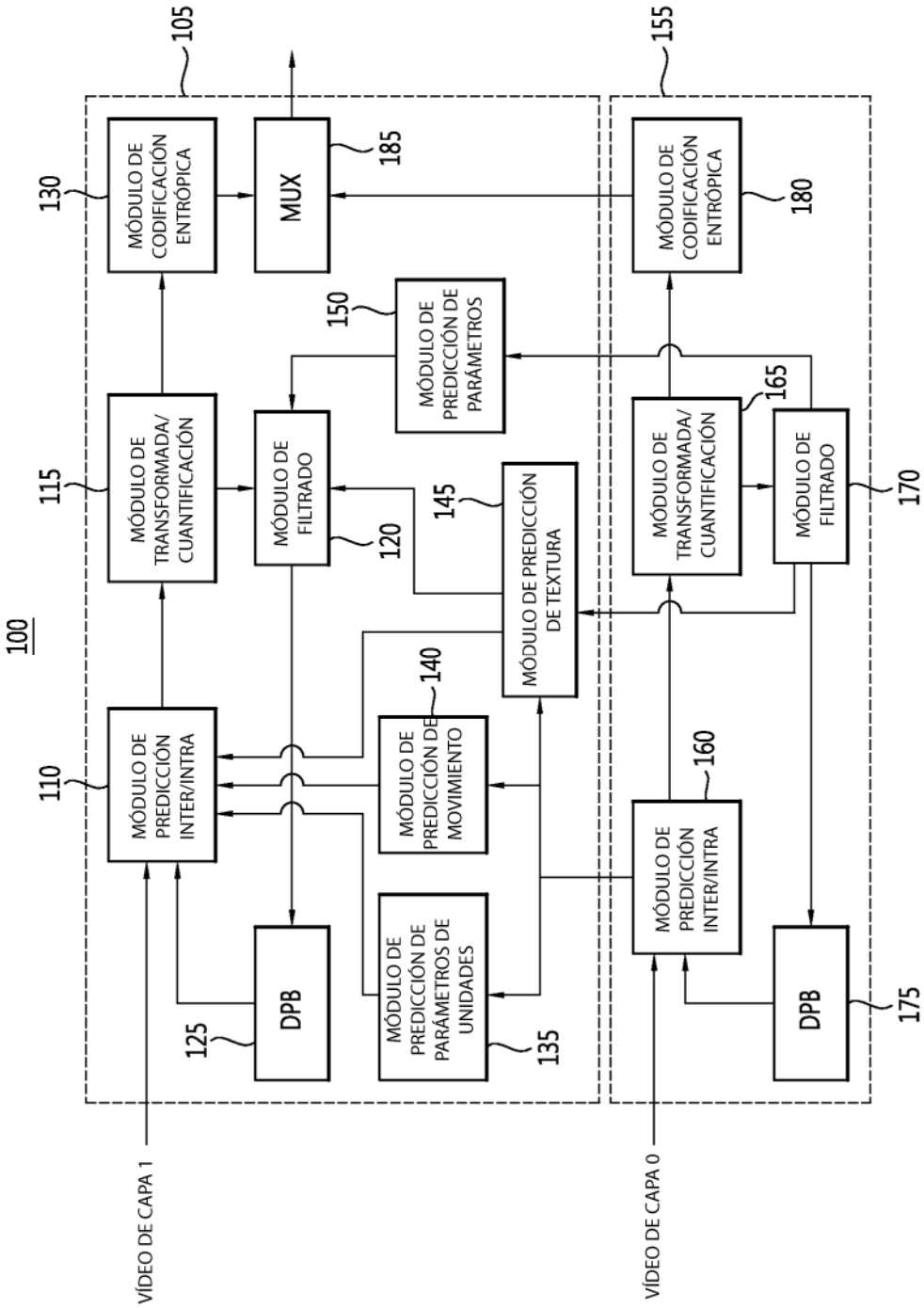

100
105
155
MÓDULO DE CODIFICACIÓN ENTRÓPICA
130
MUX
185
MÓDULO DE CODIFICACIÓN ENTRÓPICA
180
150
MÓDULO DE PREDICCIÓN DE PARÁMETROS
165
MÓDULO DE TRANSFORMADA/ CUANTIFICACIÓN
115
MÓDULO DE FILTRADO
120
145
MÓDULO DE PREDICCIÓN DE TEXTURA
MÓDULO DE TRANSFORMADA/ CUANTIFICACIÓN
MÓDULO DE FILTRADO
170
140
MÓDULO DE PREDICCIÓN INTER/INTRA
110
MÓDULO DE PREDICCIÓN DE MOVIMIENTO
160
MÓDULO DE PREDICCIÓN INTER/INTRA
DPB
125
MÓDULO DE PREDICCIÓN DE PARÁMETROS DE UNIDADES
135
DPB
175
VÍDEO DE CAPA 1
VÍDEO DE CAPA 0

FIG. 2

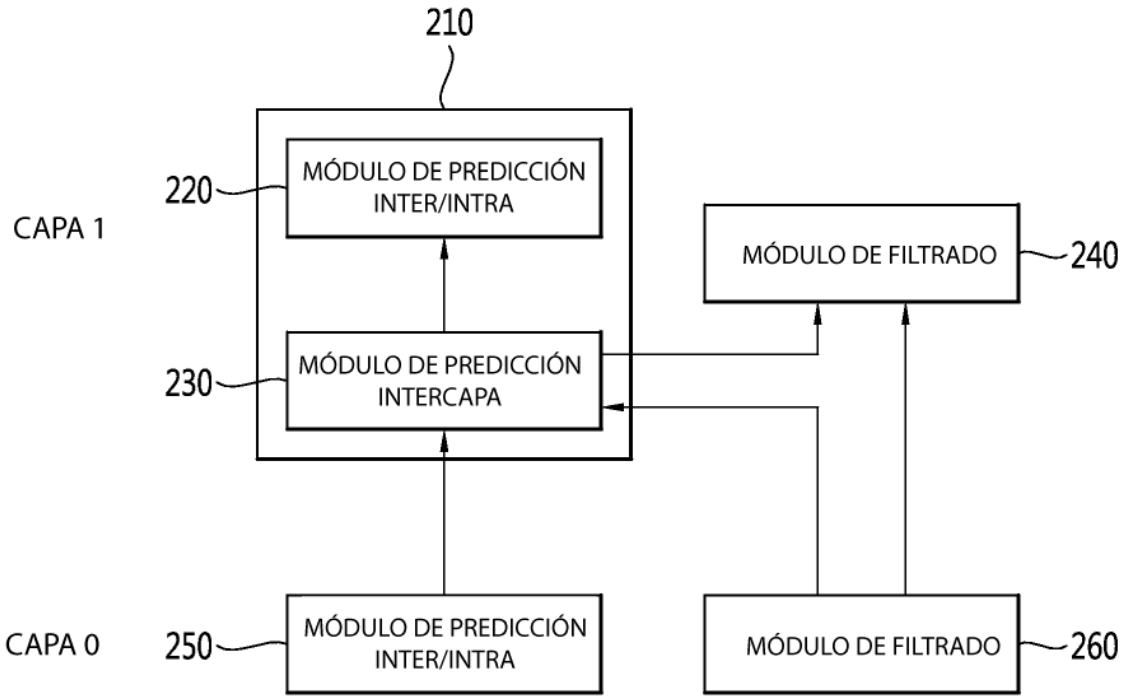

210
CAPA 1
220
MÓDULO DE PREDICCIÓN INTER/INTRA
MÓDULO DE FILTRADO
240
230
MÓDULO DE PREDICCIÓN INTERCAPA
CAPA 0
250
MÓDULO DE PREDICCIÓN INTER/INTRA
MÓDULO DE FILTRADO
260

FIG. 3

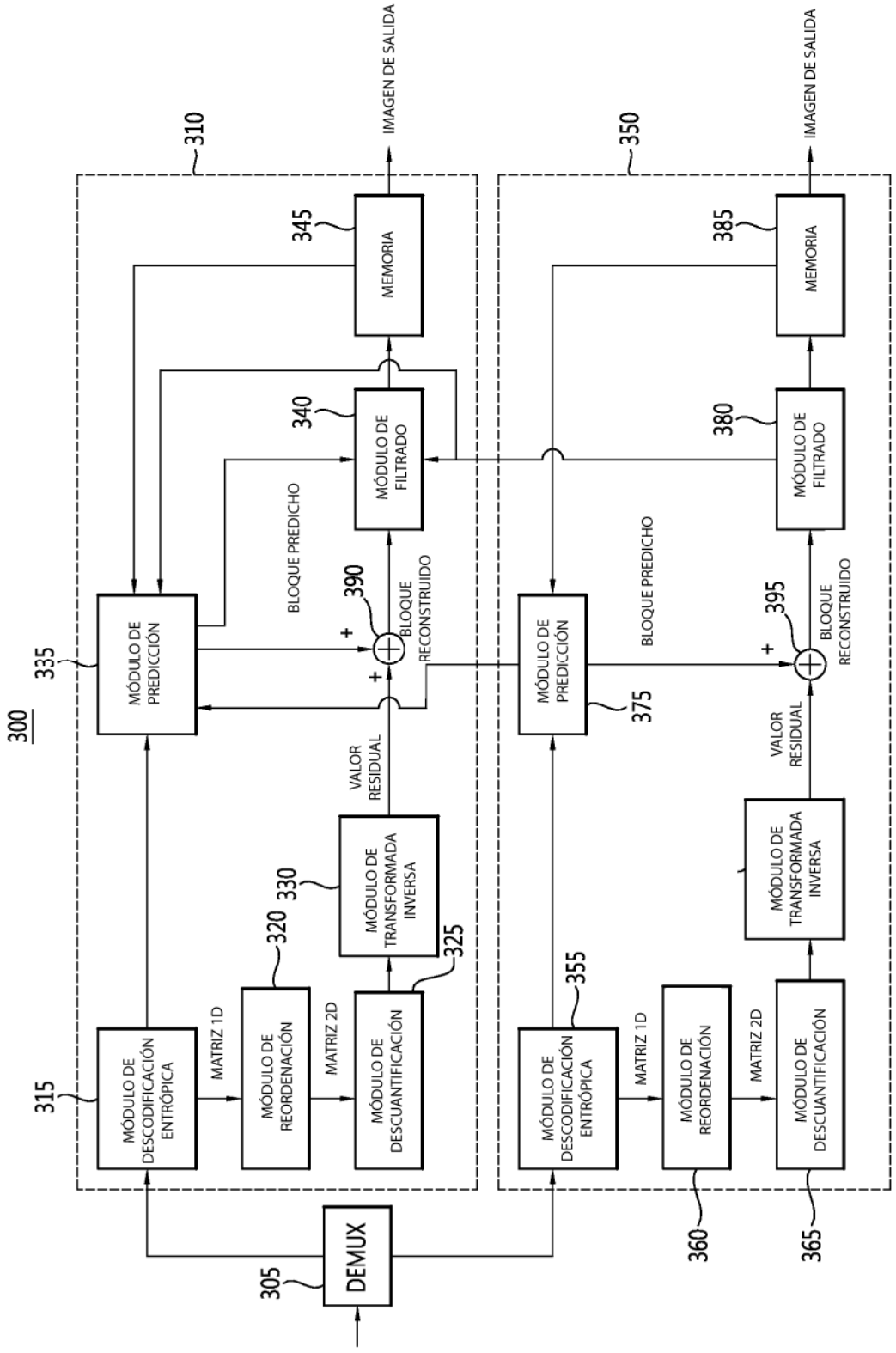

300
305
DEMUX
310
315
MÓDULO DE DESCODIFICACIÓN ENTRÓPICA
MATRIZ 1D
320
MÓDULO DE REORDENACIÓN
MATRIZ 2D
325
MÓDULO DE DESCUANTIFICACIÓN
330
MÓDULO DE TRANSFORMADA INVERSA
VALOR RESIDUAL
335
MÓDULO DE PREDICCIÓN
BLOQUE PREDICHO
340
MÓDULO DE FILTRADO
345
MEMORIA
390
BLOQUE RECONSTRUIDO
IMAGEN DE SALIDA
350
355
MÓDULO DE DESCODIFICACIÓN ENTRÓPICA
MATRIZ 1D
360
MÓDULO DE REORDENACIÓN
MATRIZ 2D
365
MÓDULO DE DESCUANTIFICACIÓN
375
MÓDULO DE PREDICCIÓN
BLOQUE PREDICHO
380
MÓDULO DE FILTRADO
385
MEMORIA
395
BLOQUE RECONSTRUIDO
VALOR RESIDUAL
MÓDULO DE TRANSFORMADA INVERSA
IMAGEN DE SALIDA

FIG. 4

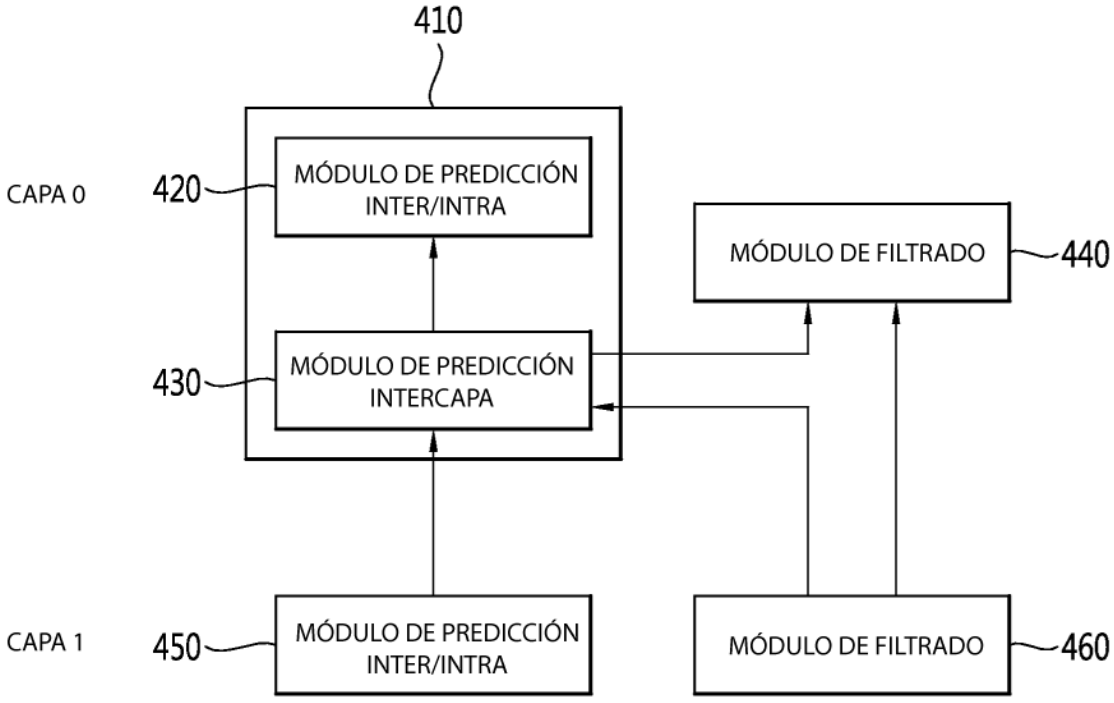

410
CAPA 0
420
MÓDULO DE PREDICCIÓN INTER/INTRA
MÓDULO DE FILTRADO
440
430
MÓDULO DE PREDICCIÓN INTERCAPA
CAPA 1
450
MÓDULO DE PREDICCIÓN INTER/INTRA
MÓDULO DE FILTRADO
460

FIG. 5

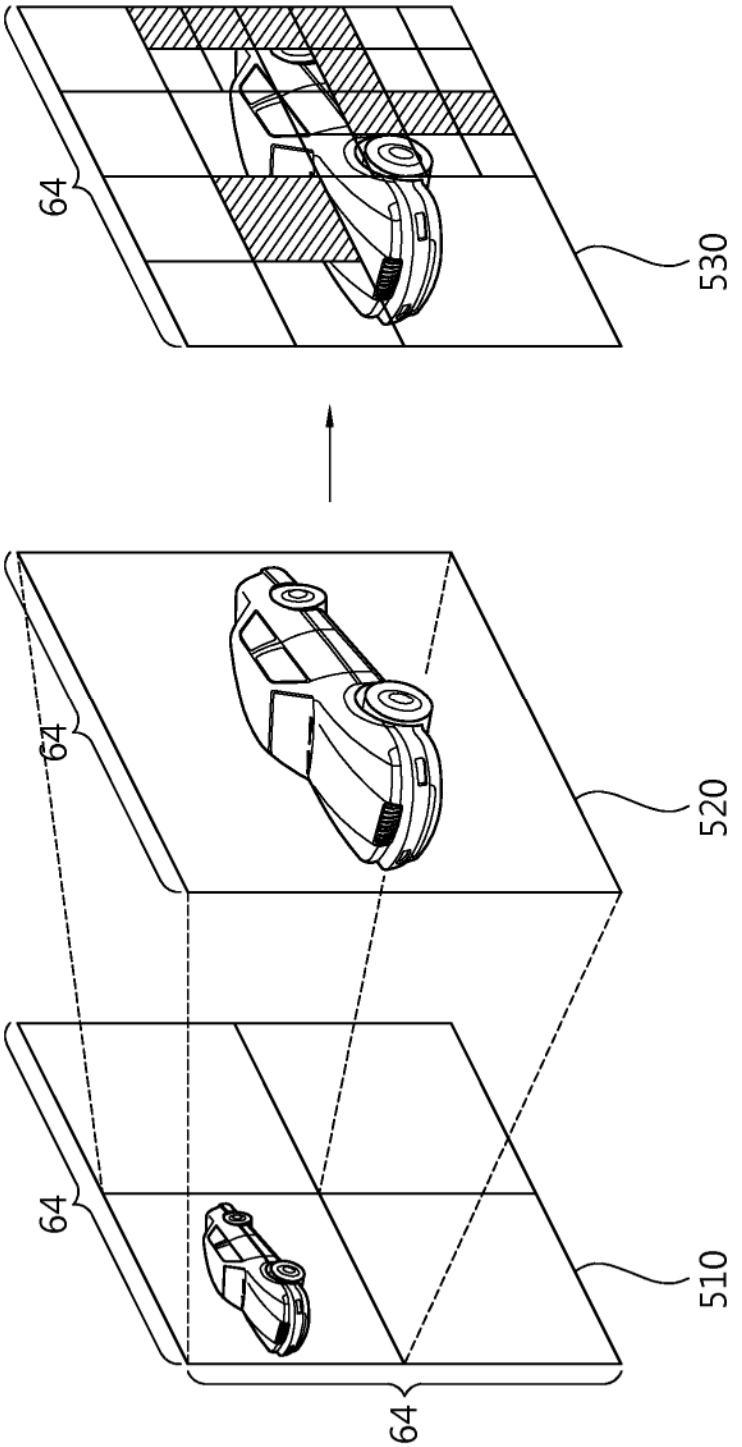


FIG. 6

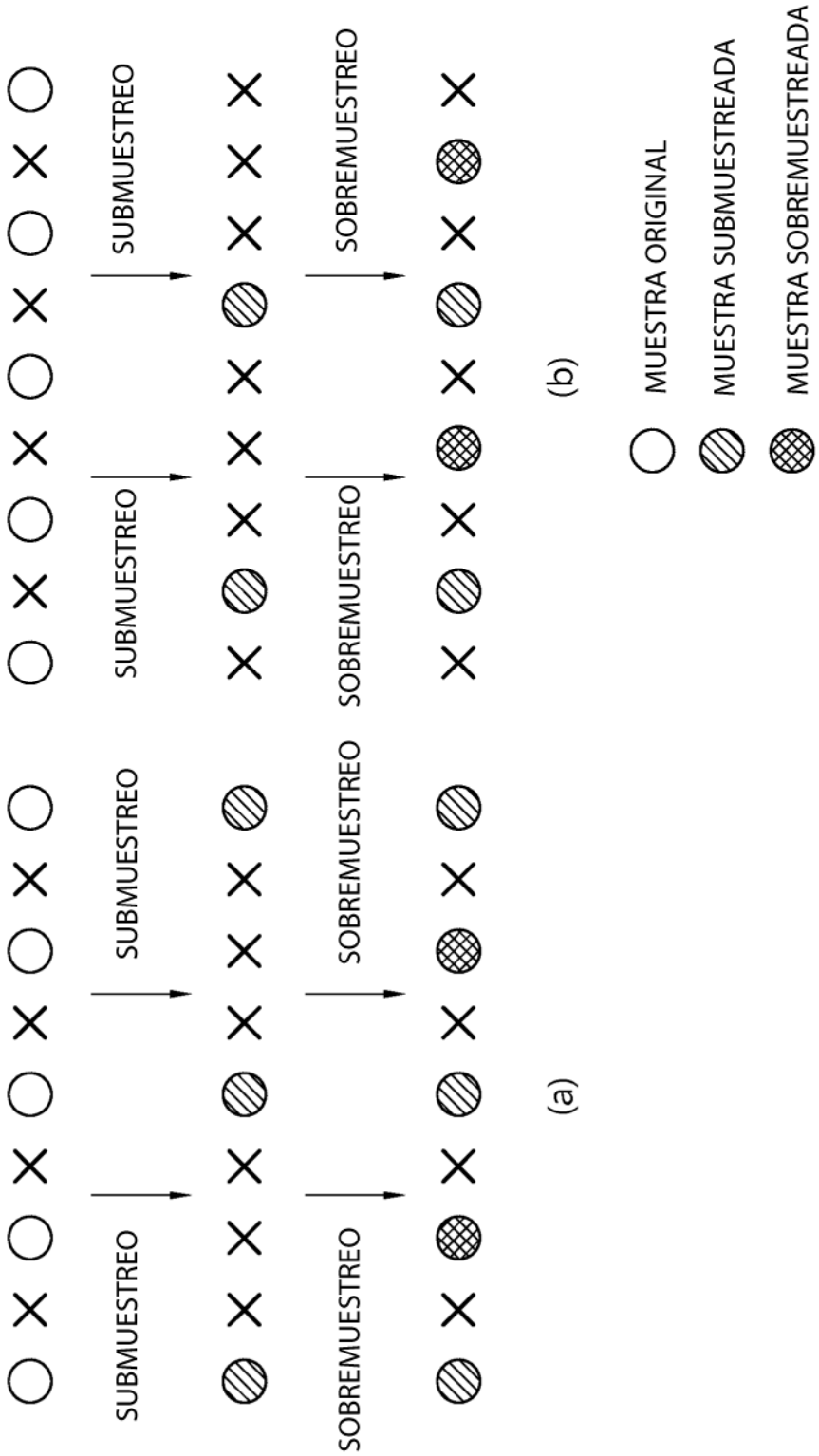

SUBMUESTREO
SUBMUESTREO
SUBMUESTREO
SOBREMUESTREO
SOBREMUESTREO
SOBREMUESTREO
(a)
(b)
MUESTRA ORIGINAL
MUESTRA SUBMUESTREADA
MUESTRA SOBREMUESTREADA

FIG. 7

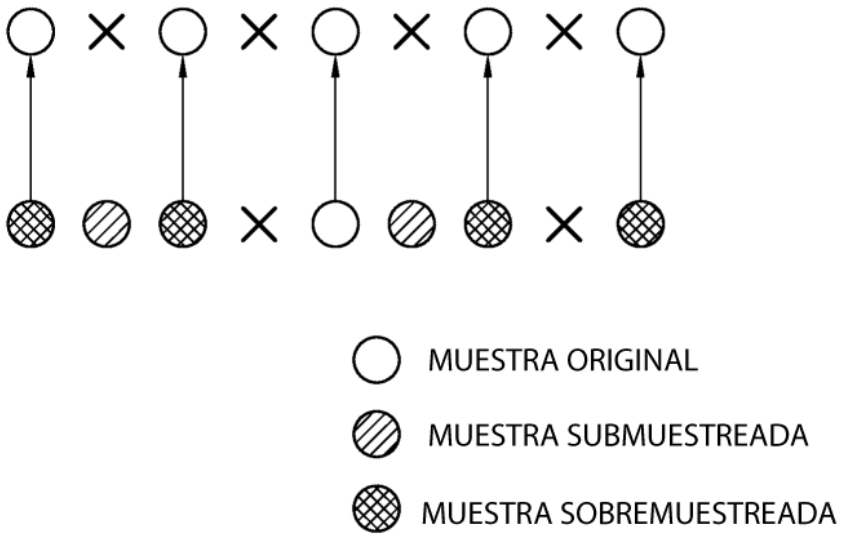

MUESTRA ORIGINAL
MUESTRA SUBMUESTREADA
MUESTRA SOBREMUESTREADA

FIG. 8

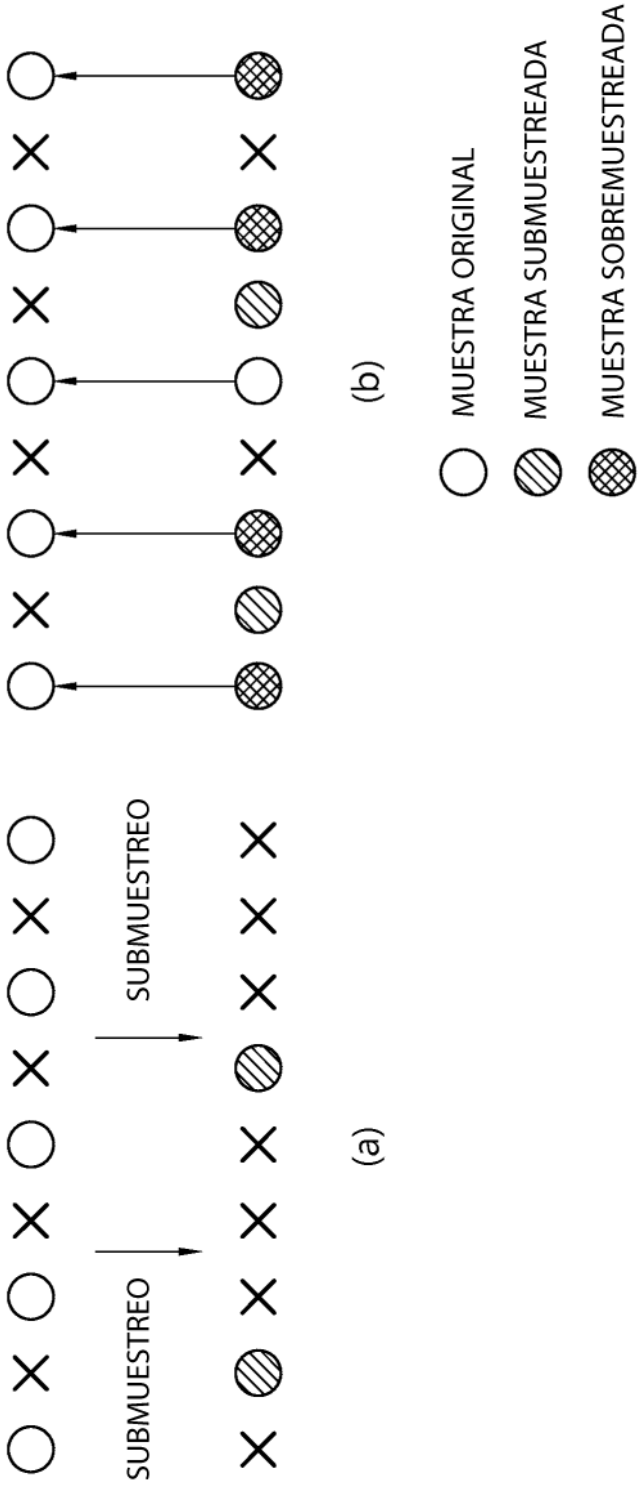

SUBMUESTREO
SUBMUESTREO
(a)
(b)
MUESTRA ORIGINAL
MUESTRA SUBMUESTREADA
MUESTRA SOBREMUESTREADA

FIG. 9

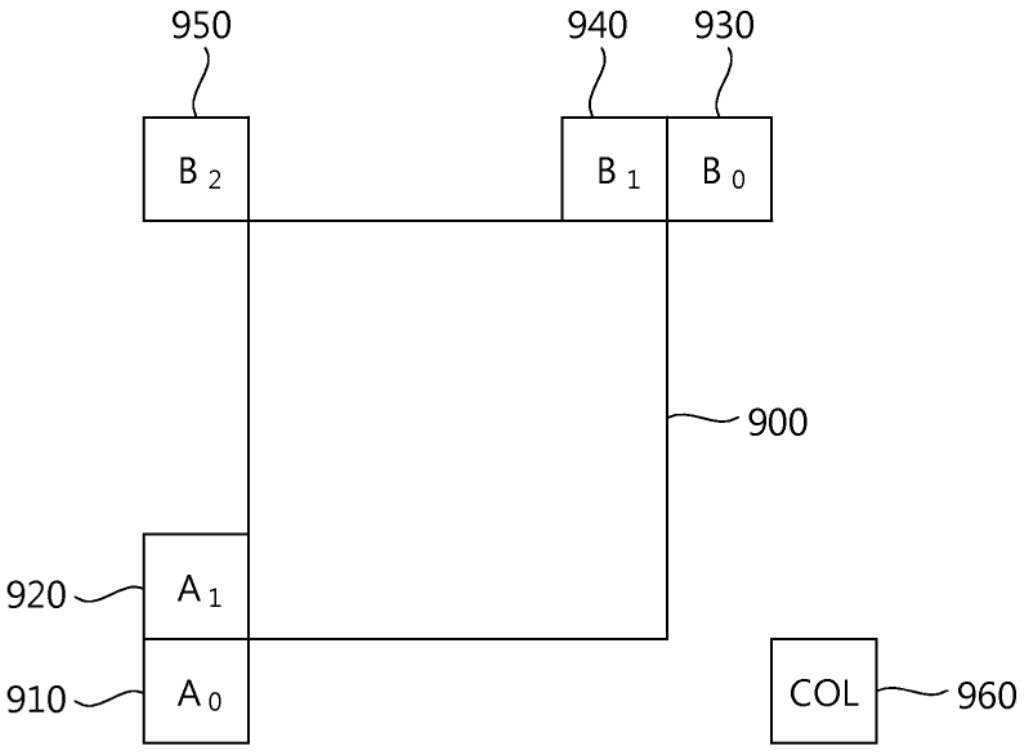


FIG. 10

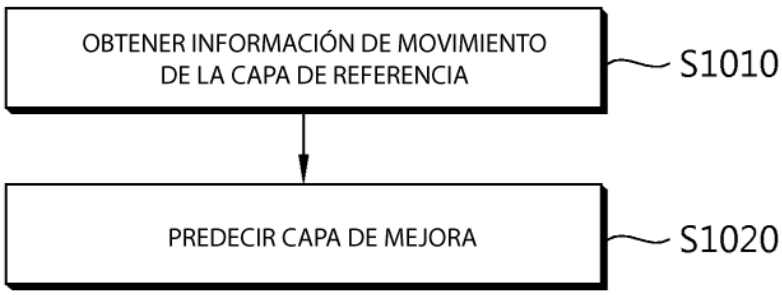

OBTENER INFORMACIÓN DE MOVIMIENTO DE LA CAPA DE REFERENCIA
S1010
PREDECIR CAPA DE MEJORA
S1020

FIG. 11

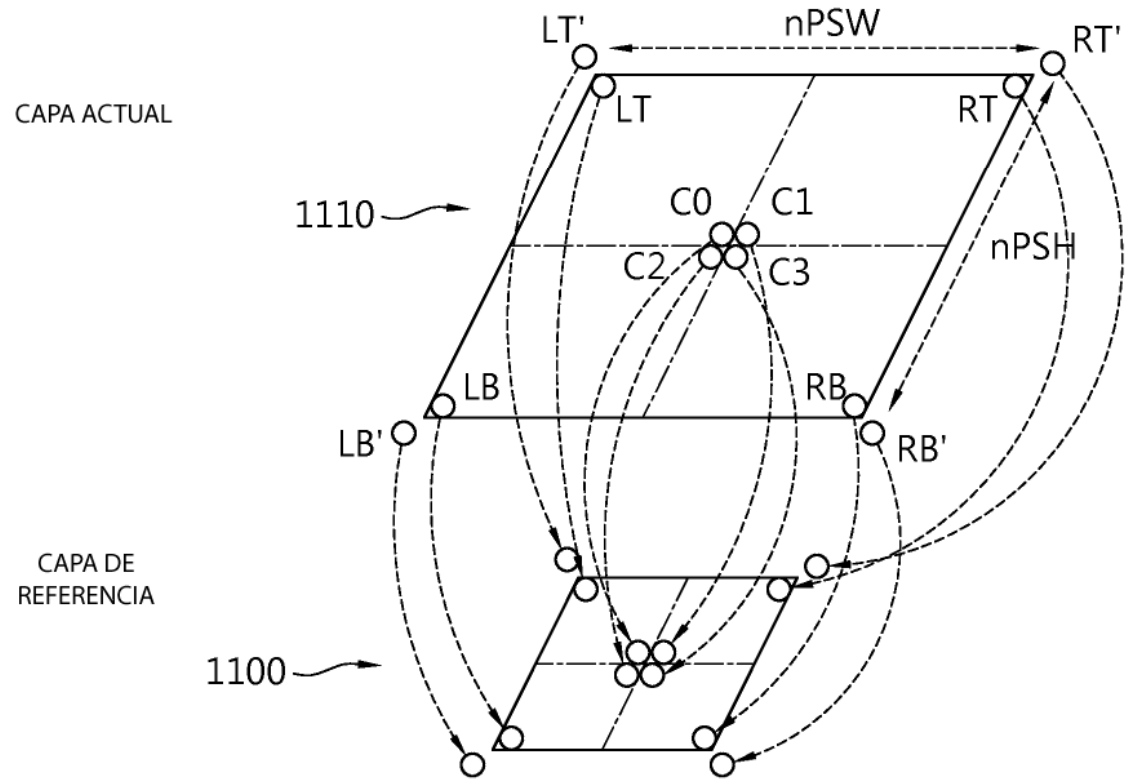

LT'
nPSW
RT'
CAPA ACTUAL
LT
RT
1110
C0
C1
C2
C3
nPSH
LB
RB
LB'
RB'
CAPA DE
REFERENCIA
1100

FIG. 12

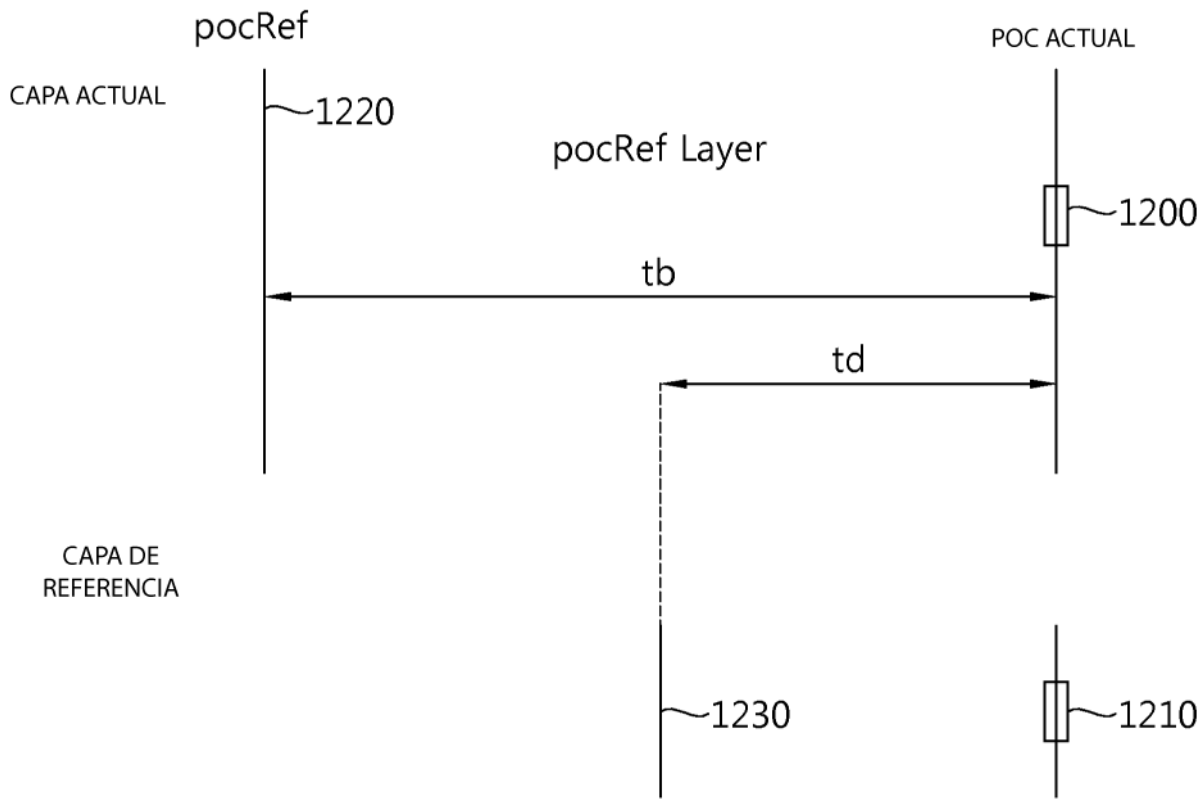

pocRef
POC ACTUAL
CAPA ACTUAL
1220
pocRef Layer
1200
tb
td
CAPA DE
REFERENCIA
1230
1210

FIG. 13

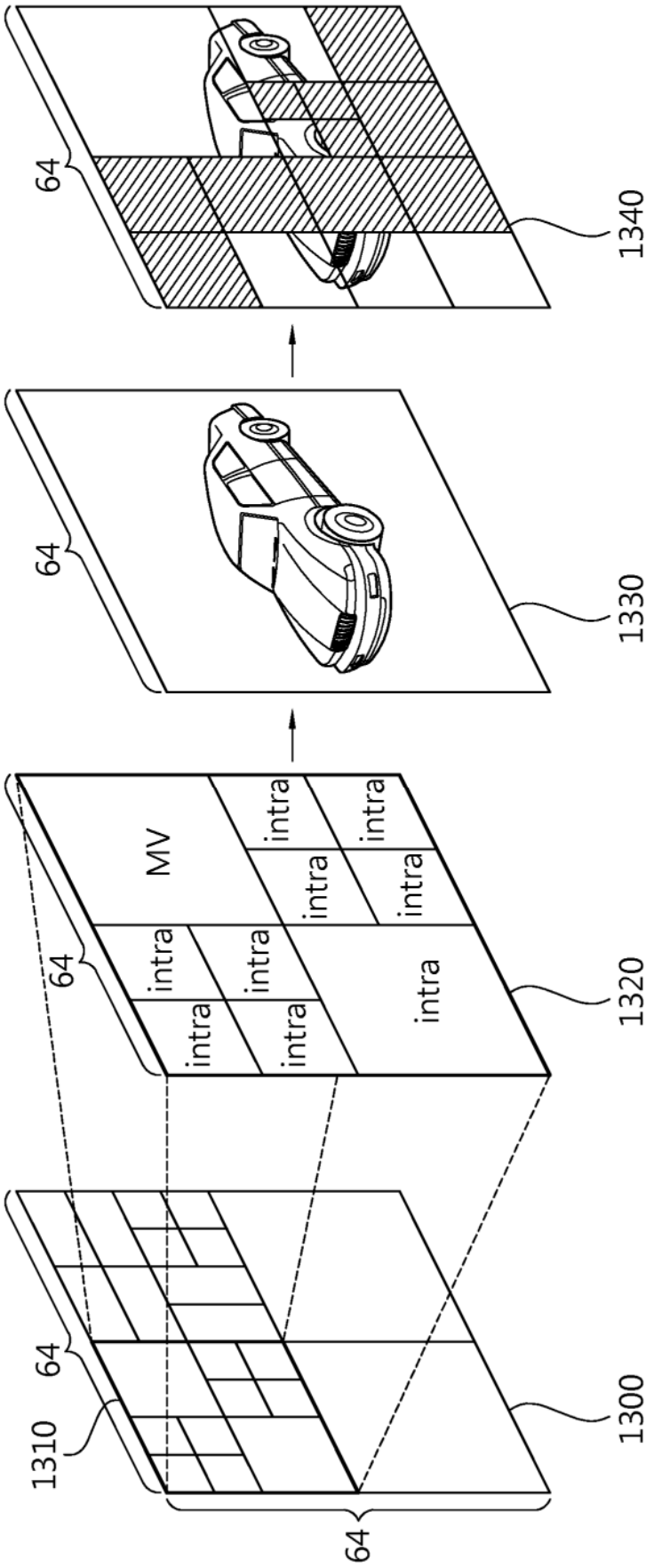

64
1340
64
1330
MV
intra
intra
intra
intra
intra
intra
intra
intra
intra
64
1320
64
1310
64
1300

FIG. 14

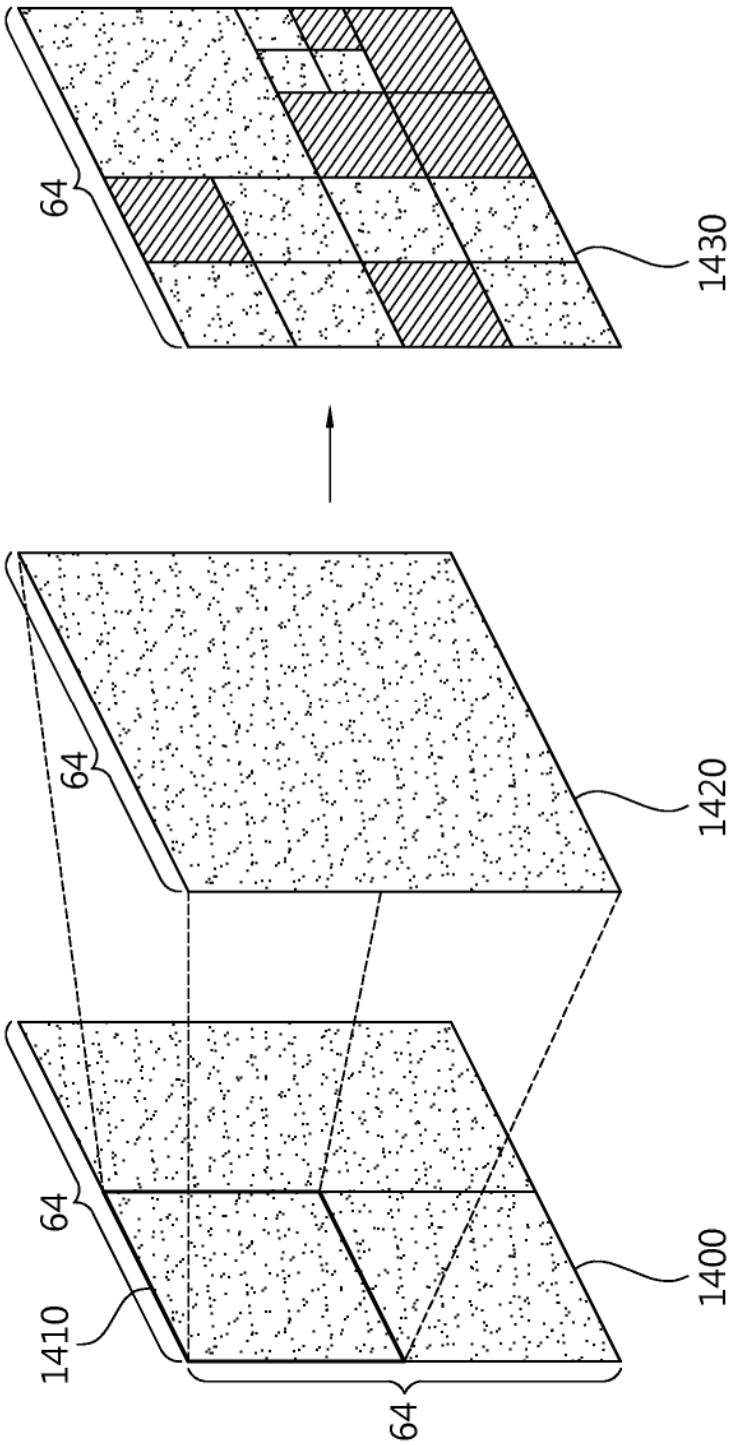


FIG. 15

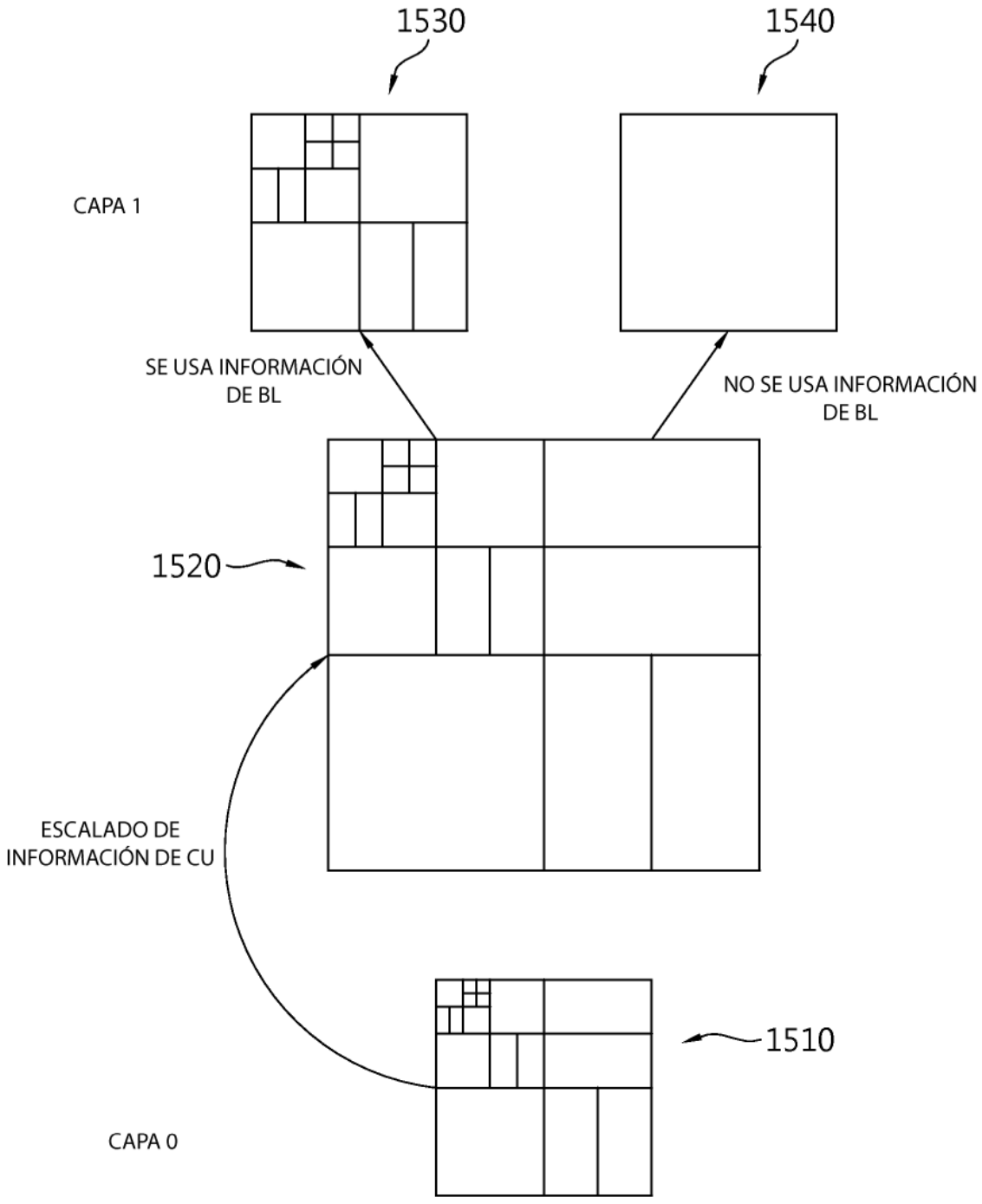

1530
1540
CAPA 1
SE USA INFORMACIÓN DE BL
NO SE USA INFORMACIÓN DE BL
1520
ESCALADO DE INFORMACIÓN DE CU
1510
CAPA 0

FIG. 16

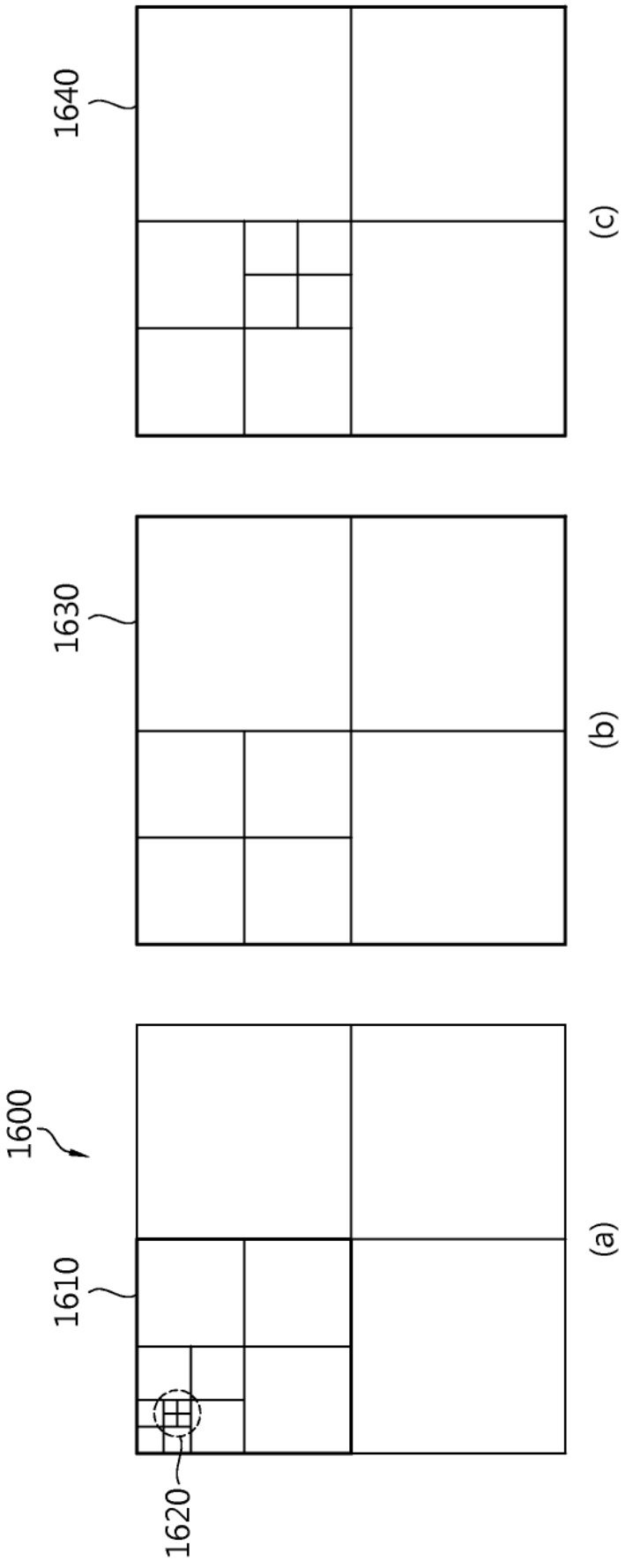


FIG. 17

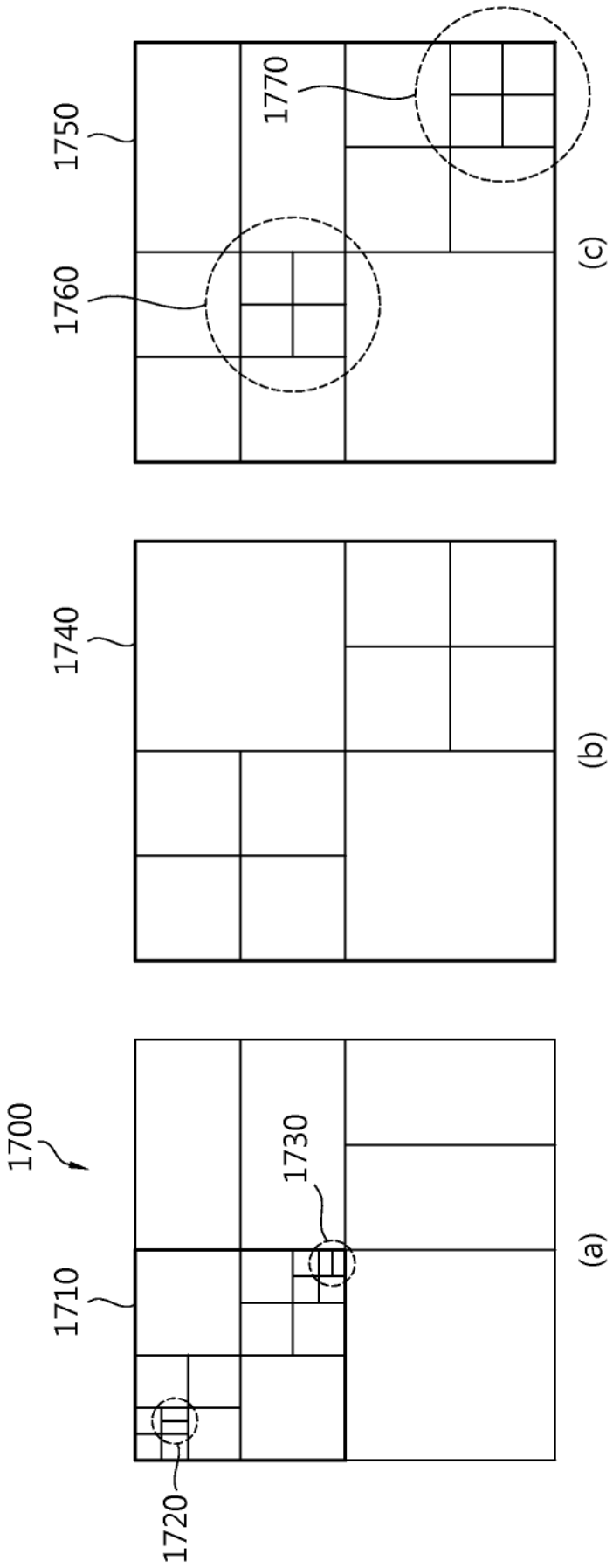


FIG. 18

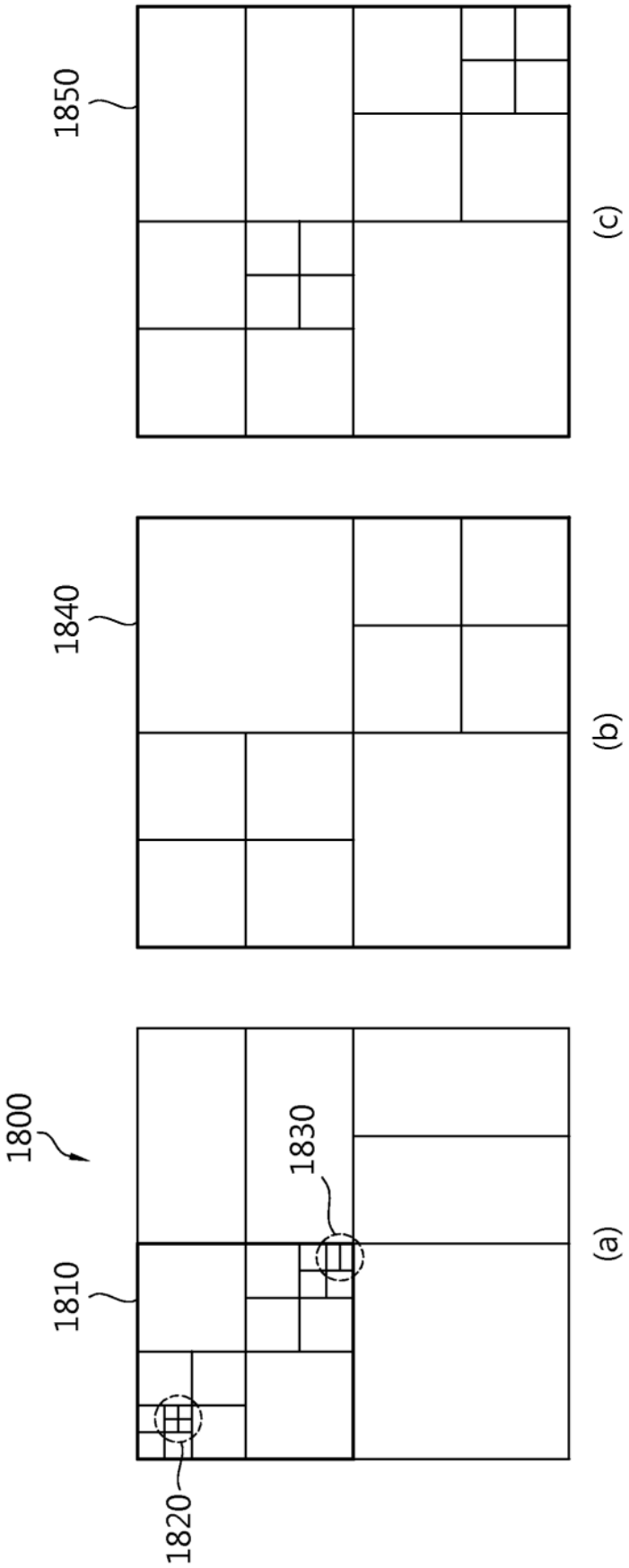


FIG. 19

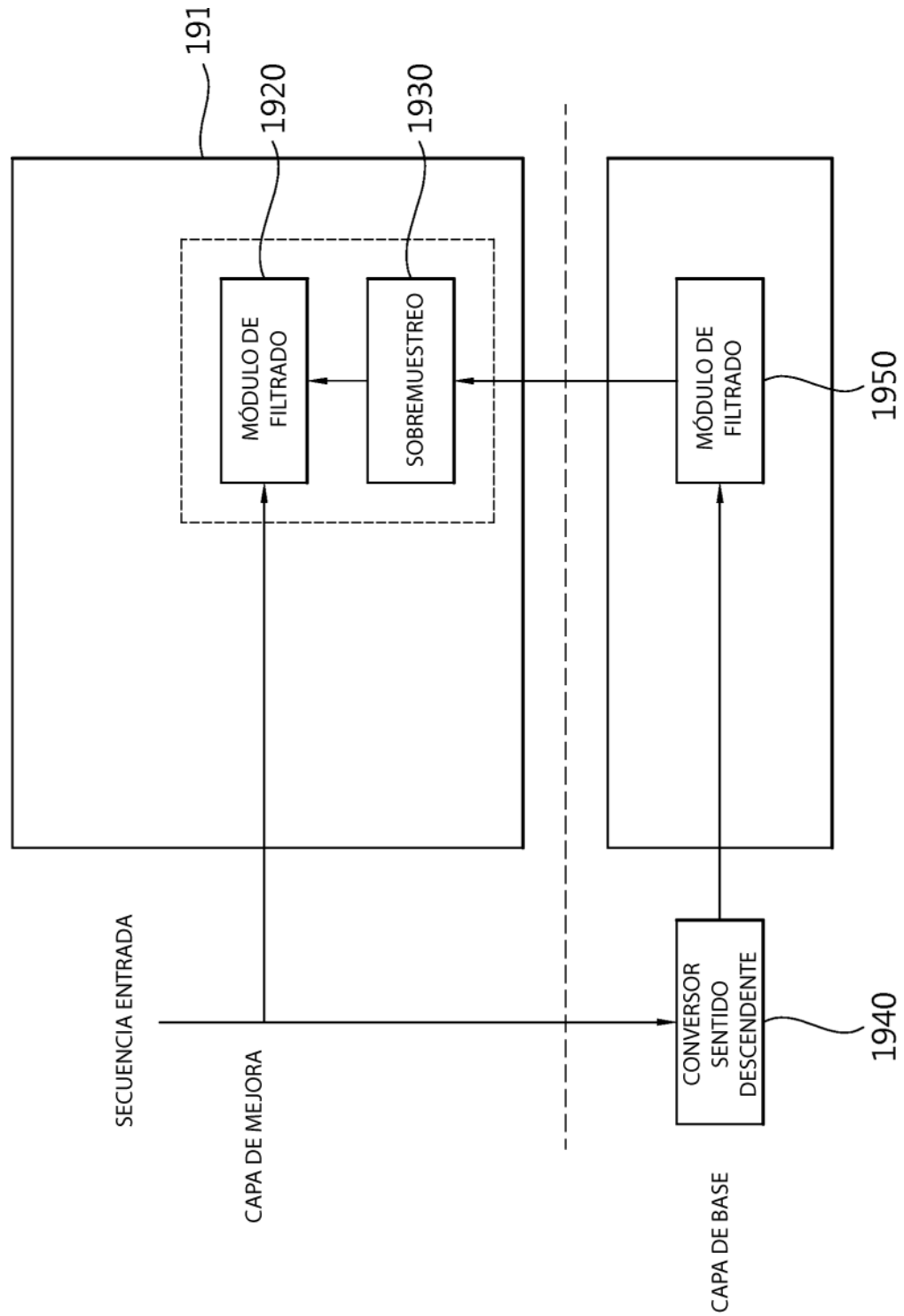

191
1920
1930
MÓDULO DE FILTRADO
SOBREMUESTREO
MÓDULO DE FILTRADO
1950
SECUENCIA ENTRADA
CONVERSOR SENTIDO DESCENDENTE
1940
CAPA DE MEJORA
CAPA DE BASE

FIG. 20

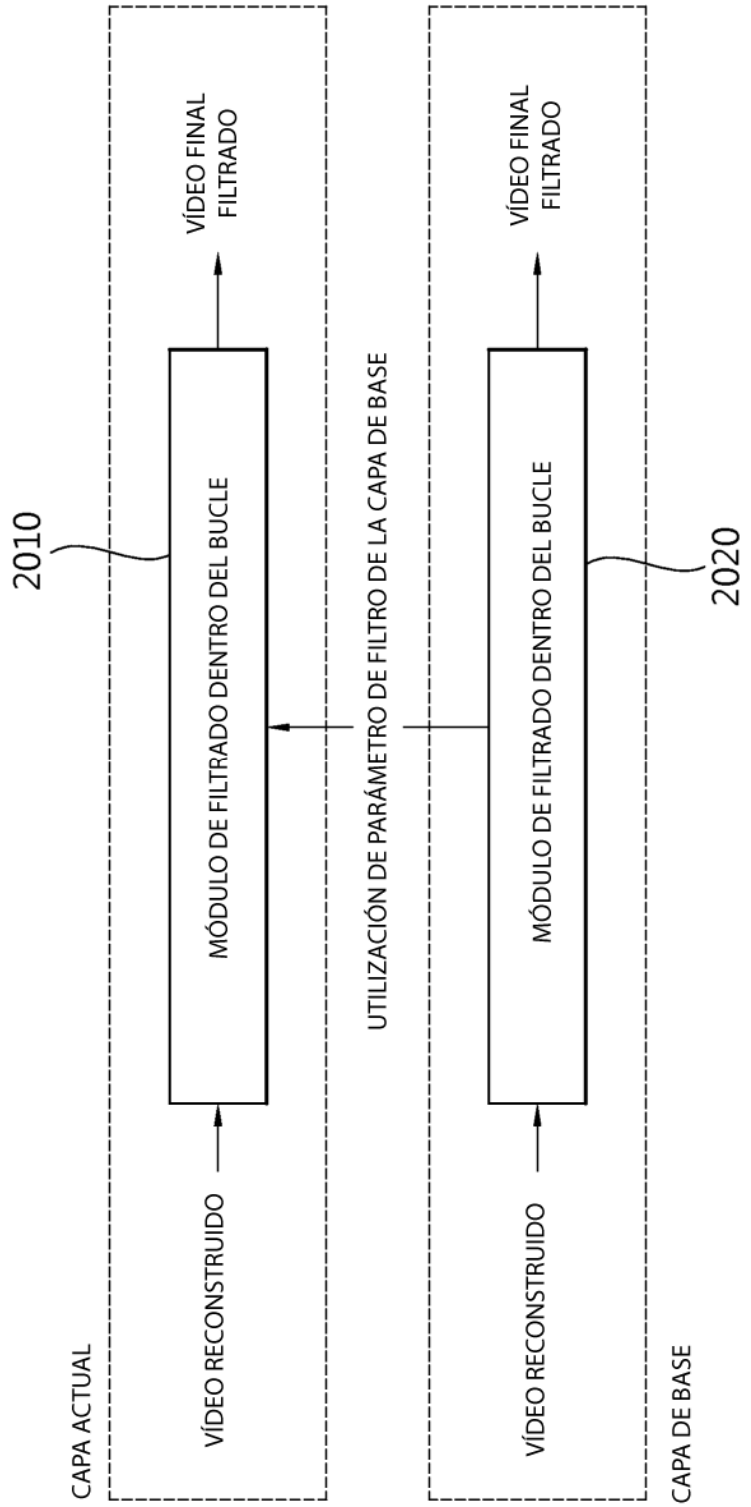
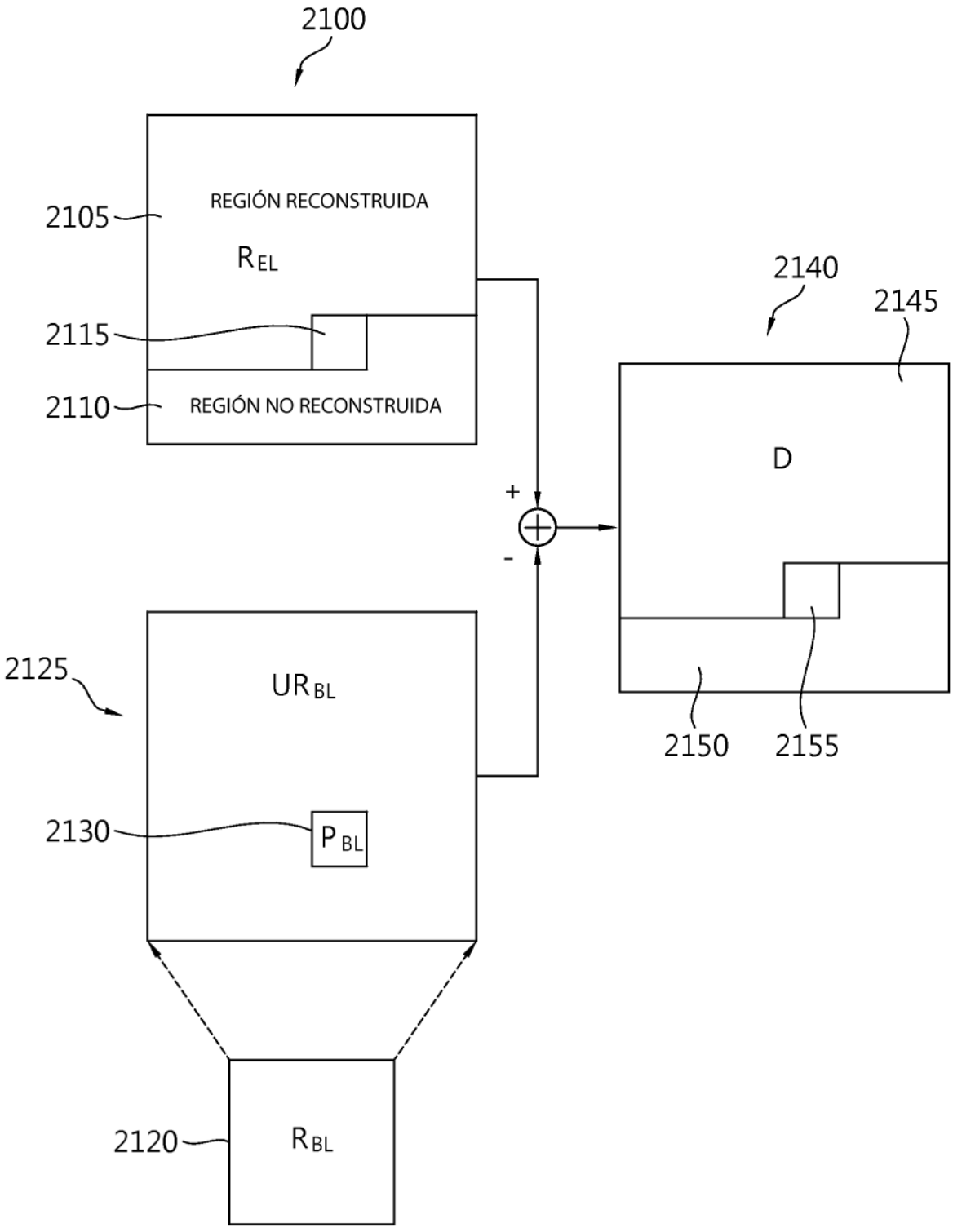

2010
2020
CAPA ACTUAL
CAPA DE BASE
VÍDEO RECONSTRUIDO
MÓDULO DE FILTRADO DENTRO DEL BUCLE
VÍDEO FINAL FILTRADO
UTILIZACIÓN DE PARÁMETRO DE FILTRO DE LA CAPA DE BASE
VÍDEO RECONSTRUIDO
MÓDULO DE FILTRADO DENTRO DEL BUCLE
VÍDEO FINAL FILTRADO

FIG. 21


2100
2105
REGIÓN RECONSTRUIDA
R_{EL}
2115
2110
REGIÓN NO RECONSTRUIDA
2140
2145
D
+
-
2125
UR_{BL}
2130
P_{BL}
2150
2155
2120
R_{BL}