(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 711 180**

(51) Int. CI.:

H04N 19/70 (2014.01)
H04N 19/117 (2014.01)
H04N 19/463 (2014.01)
H04N 19/82 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **20.04.2012 E 15151258 (9)**

(97) Fecha y número de publicación de la concesión europea: **14.11.2018 EP 2882190**

(54) Título: **Procedimiento y aparato para un filtrado en bucle mejorado**

(30) Prioridad:

21.04.2011 US 201161477689 P
14.10.2011 US 201161547281 P
07.02.2012 US 201261595914 P
07.02.2012 US 201261595900 P
13.02.2012 US 201261597995 P
17.02.2012 US 201261600028 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
30.04.2019

(73) Titular/es:

HFI INNOVATION INC. (100.0%)
3F.-7, No.5, Taiyuan 1st St.
Zhubei City, Hsinchu County 302, TW

(72) Inventor/es:

FU, CHIH-MING;
CHEN, CHING-YEH;
TSAI, CHIA-YANG;
HUANG, YU-WEN y
LEI, SHAW-MIN

(74) Agente/Representante:

CONTRERAS PÉREZ, Yahel

DESCRIPCIÓN

Procedimiento y aparato para un filtrado en bucle mejorado

La presente invención reivindica prioridad de la solicitud de patente provisional de EE.UU., número de serie 61/477.689, presentada el 21 de abril de 2011, titulada "Improved Sample Adaptive Offset", la solicitud de patente provisional de EE.UU., número de serie 61/547.281, presentada el 14 de octubre de 2011, titulada "Low Latency Loop Filtering", la solicitud de patente provisional de EE.UU., número de serie 61/595.900, presentada el 17 de febrero de 2012, titulada "Improved Sample Adaptive Offset", la solicitud de patente provisional de EE.UU., número de serie 61/595.914, presentada el 7 de febrero de 2012, titulada "Improved LCU-based Encoding Algorithm of ALF", la solicitud de patente provisional de EE.UU., número de serie 61/597.995, presentada el 13 de febrero de 2012, titulada "Improved ALF and SAO ", y la solicitud de patente provisional de EE.UU., número de serie 61/600.028, presentada el 17 de febrero de 2012, titulada "LCU-based Syntax for SAO and ALF".

CAMPO TÉCNICO

La presente invención se refiere a un sistema de codificación de video. En particular, la presente invención se refiere a un procedimiento y a un aparato para un procesamiento en bucle, tal como SAO y ALF, mejorado. Más particularmente, la presente invención se refiere a procedimientos para decodificación y codificación de video. Dichos procedimientos son conocidos de C-M FU ET AL: "Sample Adaptive Offset with LCU-Independent Decoding", 20110310, no. JCTVC-E049, 10 de marzo de 2011.

ANTECEDENTES

La estimación de movimiento es una técnica efectiva de codificación entre fotogramas para explotar la redundancia temporal en secuencias de video. La codificación entre fotogramas compensada por movimiento se ha utilizado ampliamente en diversos estándares internacionales de codificación de video. La estimación de movimiento adoptada en diversos estándares de codificación es a menudo una técnica basada en bloques, en el que se determina información de movimiento, tal como un modo de codificación y un vector de movimiento, para cada macrobloque o configuración de bloques similar. Además, la intra-codificación también se aplica de forma adaptativa, procesándose la imagen sin referenciar a ninguna otra imagen. Los residuos inter-pronosticados o intra-pronosticados generalmente son procesados adicionalmente por transformación, cuantificación y codificación de entropía para generar un flujo de bits de video comprimido. Durante el proceso de codificación, se introducen artefactos de codificación, en particular.

En la figura 1B se muestra un decodificador correspondiente para el codificador de la figura 1A. El flujo de bits de video es decodificado por el decodificador de video 142 para recuperar residuos transformados y cuantificados, información de SAO/ALF y otra información del sistema. En el lado del decodificador, solo se realiza compensación de movimiento (MC, motion compensation) 113 en lugar de ME/MC. El proceso de decodificación es similar al bucle de reconstrucción en el lado del codificador. Los residuos transformados y cuantificados recuperados, la información de SAO/ALF y otra información del sistema se utilizan para reconstruir los datos de video. El video reconstruido es procesado además por Filtro de Desbloqueo (DF) 130, Desplazamiento Adaptativo de muestras (SAO) 131 y Filtro en bucle Adaptativo (ALF) 132 para producir el video decodificado mejorado final.

HEVC ha adoptado ALF como un filtro en bucle sin desbloqueo para mejorar el rendimiento de la codificación. En el modelo de prueba de HEVC versión 5.0, se describe un algoritmo de codificación ALF basado en imágenes. Sin embargo, el proceso de cuantificación incluye un esquema de codificación basado en LCU o un proceso de canalización (pipeline process) basado en LCU. Para aliviar los artefactos de codificación, se ha aplicado un procesamiento adicional al video reconstruido para mejorar la calidad de imagen en los sistemas de codificación más nuevos. El procesamiento adicional es configurado a menudo en una operación en bucle para que el codificador y el decodificador puedan derivar las mismas imágenes de referencia para conseguir un mejor rendimiento del sistema.

La figura 1A ilustra un sistema de inter/intra codificación de video adaptativa de ejemplo que incorpora un procesamiento en bucle. Para la inter-predicción, la estimación de movimiento (ME)/compensación de movimiento (MC) 112 se utiliza para proporcionar unos datos de predicción basados en datos de video de otra imagen o imágenes. El conmutador 114 selecciona datos de intra-predicción 110 o de inter-predicción y los datos de predicción seleccionados son suministrados al agregador 116 para producir errores de predicción, también denominados residuos. Luego, el error de predicción es procesado por transformación (T) 118 seguido de cuantificación (Q) 120. Los residuos transformados y cuantificados son codificados por un codificador de entropía 122 para formar un flujo de bits de video correspondiente a los datos de video comprimidos. El flujo de bits asociado con los coeficientes de transformación es empaquetado entonces con información complementaria tal como de movimiento, modo y otra información asociada con el área de imagen. La información complementaria también puede ser sometida a codificación de entropía para reducir el ancho de banda requerido. En consecuencia, los datos asociados con la información complementaria son proporcionados al codificador de entropía 122 según se muestra

en la figura 1A. Cuando se utiliza un modo de predicción, también se debe reconstruir una imagen o imágenes de referencia en el extremo del codificador. En consecuencia, los residuos transformados y cuantificados son procesados por cuantificación inversa (IQ) 124 y transformación inversa (IT) 126 para recuperar los residuos. Luego, se vuelven a agregar los residuos a los datos de predicción 136 en reconstrucción (REC) 128 para reconstruir los datos de video. Los datos de video reconstruido pueden ser almacenados en el búfer de imágenes de referencia 134 y usados para la predicción de otros fotogramas.

Según se muestra en la figura 1A, los datos de video entrantes se someten a una serie de procesamientos en el sistema de codificación. Los datos de video reconstruidos por la REC 128 pueden sufrir diversas deficiencias debido a una serie de procesamientos. Por consiguiente, se aplica un procesamiento en bucle a los datos de video reconstruido antes de que los datos de video reconstruido se almacenen en el búfer de imágenes de referencia 134 con el fin de mejorar la calidad del video. En el estándar de Codificación de Video de Alta Eficiencia (HEVC) que está en evolución, se han desarrollado el Filtro de Desbloqueo (DF) 130, el Desplazamiento Adaptativo de muestras (SAO) 131 y el Filtro en bucle Adaptativo (ALF) 132 para mejorar la calidad de imagen. La información del filtro en bucle puede tener que incorporarse en el flujo de bits para que un decodificador pueda recuperar de forma adecuada la información requerida. Por lo tanto, la información de filtro en bucle de SAO y ALF es proporcionada al codificador de entropía 122 para su incorporación al flujo de bits. En la figura 1A, se aplica primero el DF 130 al video reconstruido; luego se aplica el SAO 131 al video procesado por DF; y se aplica el ALF 132 al video procesado por el SAO. Sin embargo, el orden de procesamiento entre DF, SAO y ALF puede ser reorganizado, utilizado normalmente para implementaciones de codificador y decodificador de video debido a un uso más eficiente de la memoria, menos ancho de banda de memoria o costes de hardware menores. Por lo tanto, el ALF basado en LCU es un enfoque preferido. Sin embargo, es deseable mejorar aún más el rendimiento del procesamiento del ALF.

El SAO es otro procesamiento en bucle adoptado por el HEVC para mejorar la calidad de imagen. El SAO consta de dos procedimientos. Uno de ellos es el desplazamiento de banda (BO, Band Offset), y el otro es el desplazamiento de borde (EO, Edge Offset). El BO se utiliza para clasificar los píxeles en múltiples bandas según las intensidades de los píxeles y se aplica a los píxeles un desplazamiento en cada banda. El EO se utiliza para clasificar los píxeles en categorías según relaciones con los vecinos y se aplica un desplazamiento a los píxeles en cada categoría. En el HM-5.0, una región puede seleccionar 7 tipos diferentes de SAO: 2 grupos BO (grupo externo y grupo interno), 4 patrones direccionales de EO (0°, 90°, 135° y 45°) y ningún procesamiento (OFF). Además, una imagen puede dividirse además en múltiples regiones usando un procedimiento de división de árbol cuaternario (quad-tree) o dividirse en regiones de unidades de codificación más grandes (LCU), y teniendo cada región su propio tipo de SAO y valores de desplazamiento. Es deseable mejorar aún más el rendimiento del procesamiento de SAO mejorando la señalización de los parámetros de SAO.

El artículo "H.264/MPEG-4 Part 10 White Paper", XP 002281494, divulga que los 4 modos de predicción para croma son muy similares a los modos de predicción 16x16 de luma, excepto que el orden de los números de modo es diferente: Modo DC, modo horizontal, modo vertical y modo plano.

El documento US 2005/0053294 A1 divulga que se obtiene un vector de movimiento de croma nominal combinando y escalando los vectores de movimiento de luma de manera apropiada.

RESUMEN

Se describen procedimientos para la decodificación y codificación de video de acuerdo con la presente invención en las reivindicaciones independientes 1 y 5, respectivamente. Las respectivas reivindicaciones dependientes definen formas de realización preferidas de los mismos, respectivamente.

En la reivindicación 9 se define un flujo de bits de vídeo de acuerdo con la invención.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La figura 1A ilustra un sistema de inter/intra codificación de video adaptativa de ejemplo con un procesamiento en bucle DF, SAO y ALF.

La figura 1B ilustra un sistema de inter/intra codificación de video adaptativa de ejemplo con un procesamiento en bucle DF, SAO y ALF.

La figura 2 ilustra un ejemplo de compartición de parámetros de filtro en bucle entre un bloque actual y bloques vecinos.

La figura 3 ilustra un ejemplo de clasificación mejorada de desplazamiento de bordes basada en ocho píxeles vecinos en una ventana de 3x3 en torno al píxel actual.

La figura 4 ilustra un diseño de sintaxis de ejemplo para que un seq_parameter_set_rbsp() habilite un control adaptativo de procesamiento de SAO a través de un límite de segmento.

La figura 5 ilustra un diseño de sintaxis de ejemplo para que un aps_rbsp() habilite una señalización de parámetros de SAO adaptativa para incorporar parámetros de SAO en un APS o para intercalar con datos de bloque en un segmento.

La figura 6 ilustra un diseño de sintaxis de ejemplo para que un aps_sao_param() permita que bloques de codificación en un segmento con aps_id que referencia al APS compartan parámetros de SAO en el APS.

La figura 7 ilustra un diseño de sintaxis de ejemplo para que un sao_unit_vlc() incorpore información de SAO para bloques de codificación.

La figura 8 ilustra un diseño de sintaxis de ejemplo para que un sao_offset_vlc() incorpore valores de desplazamiento de SAO para bloques de codificación.

La figura 9 ilustra un diseño de sintaxis de ejemplo para que un slice_header() habilite un intercalado adaptativo de parámetros de SAO con datos de bloque.

La figura 10 ilustra un diseño de sintaxis de ejemplo para que unos slice_data() habiliten un intercalado adaptativo de parámetros de SAO con datos de bloque.

La figura 11 ilustra un diseño de sintaxis de ejemplo para que un sao_unit_cabac() habilite un intercalado adaptativo de parámetros de SAO con datos de bloque.

La figura 12 ilustra un diseño de sintaxis de ejemplo para que un sao_offset_cabac() habilite un intercalado adaptativo de parámetros de SAO con datos de bloque.

La figura 13 ilustra un diseño de sintaxis de ejemplo para que un seq_parameter_set_rbsp() habilite una señalización adaptativa de parámetros de ALF.

La figura 14 ilustra un diseño de sintaxis de ejemplo para que una slice_header() habilite una señalización adaptativa de parámetros de ALF.

La figura 15 ilustra un diseño de sintaxis de ejemplo para que un alf_param() habilite una señalización adaptativa de parámetros de ALF.

La figura 16 ilustra un diseño de sintaxis de ejemplo para que una alf_unit() habilite una señalización adaptativa de parámetros de ALF.

La figura 17 ilustra un diseño de sintaxis de ejemplo para que una alf_info() habilite una señalización adaptativa de parámetros de ALF.

DESCRIPCIÓN DETALLADA

En un filtrado en bucle basado en bloques, los parámetros del filtro en bucle para cada bloque tienen que ser suministrados al lado del decodificador para una correcta operación del filtrado en bucle en el lado del decodificador. Ejemplos de filtro en bucle pueden incluir desplazamiento adaptativo de muestras (SAO) o filtro en bucle adaptativo (ALF). El filtro en bucle también se denomina, en esta divulgación, procesamiento de filtro en bucle o procesamiento en bucle. Los parámetros en bucle también se denominan información en bucle en esta divulgación. Cuando el tamaño de bloque es pequeño, tal como una unidad de codificación (CU) o una unidad de codificación más grande (LCU), la tasa de bits correspondiente a los parámetros del filtro en bucle pasa a ser relativamente alta. Por lo tanto, es deseable reducir la tasa de bits asociada con los parámetros del filtro en bucle. Una forma de realización de acuerdo con la presente invención usa un indicador de fusión para indicar el caso en el que un bloque actual comparte los parámetros de filtro en bucle con un bloque vecino o con bloques vecinos. Además, el bloque puede incluir múltiples LCU para disminuir la tasa de bits correspondiente a los parámetros del filtro en bucle. El indicador de fusión también se puede determinar implícitamente. Por ejemplo, si los bloques izquierdo y superior tienen los mismos parámetros del filtro en bucle, el bloque actual compartirá los parámetros del filtro en bucle con sus vecinos izquierdo y superior sin la necesidad de transmitir explícitamente el indicador de fusión. La información del filtro en bucle se puede incorporar al flujo de bits de video comprimido.

El parámetro de filtro en bucle que se comparte para un bloque actual puede seleccionar uno de los bloques vecinos, tal como el bloque izquierdo, superior, derecho, superior izquierdo, inferior, inferior izquierdo, etc. La sintaxis de fusión se utiliza para indicar que dos bloques se fusionan y comparten los mismos parámetros del filtro en bucle.

La figura 2 ilustra un ejemplo de compartición de parámetros de filtro en bucle con un bloque vecino. A continuación se describe un proceso de ejemplo para determinar si un bloque C actual debería compartir parámetros del filtro en bucle con un bloque vecino. Si la información de SAO del bloque A no es igual a la del bloque B, X1 se establece igual a uno. De lo contrario, X1 se establece igual a cero. Si la información de SAO del bloque D no es igual a la del bloque B, X2 se establece igual a uno. De lo contrario, X2 se establece igual a cero. Si la información de SAO del bloque D no es igual a la del bloque A, X3 se establece igual a uno. De lo contrario, X3 se establece igual a cero. La variable X se puede calcular de acuerdo con X = X3*4 + X2*2 + X1*1. La variable X se puede usar para seleccionar un bloque vecino para fusionarlo con el bloque actual.

En un proceso de fusión de ejemplo, cuando X es igual a 7, se incorpora un indicador de fusión para indicar si el bloque C usa o no usa nuevos parámetros del filtro en bucle. Cuando el indicador de fusión tiene un valor igual a uno, el bloque C utiliza nuevos parámetros del filtro en bucle y, de lo contrario, el bloque C comparte los parámetros del filtro en bucle con el bloque A o B. Cuando X es igual a 5, se incorpora un indicador de fusión para indicar si el bloque C usa o no usa nuevos parámetros del filtro en bucle. Cuando el indicador de fusión tiene un valor igual a uno, el bloque C utiliza nuevos parámetros del filtro en bucle y, de lo contrario, el bloque C comparte los parámetros del filtro en bucle con el bloque A. Cuando X es igual a 3, se incorpora un indicador de fusión para indicar si el bloque C usa o no usa nuevos parámetros del filtro en bucle. Cuando el indicador de fusión tiene un valor igual a uno, el bloque C utiliza nuevos parámetros del filtro en bucle y, de lo contrario, el bloque C comparte los parámetros del filtro en bucle con el bloque B. Cada bloque puede ser una unidad de codificación, varias unidades de codificación, una unidad de codificación más grande, varias unidades de codificación más grandes, u otra estructura de bloques. El bloque también puede corresponder a diferentes tamaños. El bloque también se denomina bloque de codificación en esta divulgación.

En un enfoque convencional, el procesamiento de SAO sólo se aplica al componente de luma. Una forma de realización de acuerdo con la presente invención también aplica selectivamente el procesamiento de SAO a los componentes de croma si el procesamiento de SAO se aplica al componente de luma. Cuando el procesamiento de SAO se aplica al componente de luma, se puede usar un indicador de croma de SAO para indicar si el procesamiento de SAO se aplica a los componentes de croma. La información de SAO para un componente de croma puede ser derivada a partir de la información de SAO para el componente de luma u otro(s) componente(s) de croma. Cuando un componente de croma comparte la información de SAO con el componente de luma u otro(s) componente(s) de croma, se puede usar un indicador de compartición de información de SAO para indicar el caso. La información de SAO puede incluir valores de desplazamiento, tipo de SAO (también denominado categoría o clase en esta divulgación) y decisión activar/desactivar (ON/OFF). En una forma de realización, los valores de desplazamiento son predefinidos de manera que se puede usar un índice para seleccionar uno de los valores de desplazamiento. La información de SAO puede ser incorporada en una carga útil de datos en el flujo de bits tal como un PPS, un APS o una cabecera de segmento a compartir por múltiples bloques.

La información asociada con los parámetros de SAO puede codificarse en base a parámetros de SAO de un bloque procesado anteriormente utilizando un procedimiento predictivo. La codificación del valor de desplazamiento puede depender del aumento interno de la profundidad de bits, información de modo inter/intra, información de movimiento, tamaño de transformación, tamaño de etapa de cuantificación, información de desbloqueo, residual, tamaño de imagen y tamaño de región. Por ejemplo, si el aumento interno de la profundidad de bits es mayor que uno, se puede aumentar en un bit la codificación del valor de desplazamiento. En otro ejemplo de codificación de información de SAO, los parámetros de SAO pueden cambiar su precisión de codificación de acuerdo con los parámetros de cuantificación. Con el fin de reducir la tasa de bits asociada con los parámetros de SAO, se puede codificar la información para los valores de desplazamiento de un modo de tiempo retardado. Además, la información de SAO de un segmento actual puede compartirse con otros segmentos o regiones. En consecuencia, se puede incorporar la información de desplazamiento al conjunto de parámetros de imagen (PPS). Los bloques de codificación de un segmento o región actual pueden compartir la misma información de SAO.

En un enfoque convencional, el procesamiento de SAO se aplica a menudo después del procesamiento de DF. Una forma de realización de acuerdo con la presente invención puede aplicarse selectivamente entre el procesamiento de DF y el procesamiento de SAO a un bloque. Alternativamente, tanto el procesamiento de DF como el procesamiento de SAO pueden aplicarse a los mismos datos de video reconstruido y las salidas de ambos procesamientos son combinadas linealmente.

Según una forma de realización de la presente invención, el procedimiento de clasificación de píxeles para el SAO se puede combinar con otros procedimientos de clasificación de píxeles tales como dirección de borde, intensidad de píxeles, variación de píxeles, varianza de píxeles, suma de píxeles de Laplace, resultado de un filtrado de paso alto, resultado de un filtrado de paso bajo, valor absoluto del resultado del filtrado de paso alto, y valor medio de píxeles vecinos. Por ejemplo, una categoría de desplazamiento de banda (BO) o desplazamiento de borde (EO) puede ser dividida aún más por el EO. En otro ejemplo, una categoría de EO puede ser dividida aún más por el BO o suma de Laplace.

En la clasificación de EO convencional, se utilizan dos píxeles vecinos en una ventana de 3x3 para clasificar un píxel actual en diferentes categorías o clases. Una forma de realización de acuerdo con la presente invención puede usar un procedimiento de clasificación mejorado basado en todos los píxeles vecinos en la ventana de 3x3. Los píxeles vecinos (P1 - P8) en torno al píxel actual C se muestran en la figura 3. Como un ejemplo, el índice de clase, ClassIdx se puede definir como:

$$ClassIdx = Index2ClassTable(f(C,P_1) + f(C,P_2) + ... + f(C,P_8)),$$

en la que $f(C,P_n)$ es una función de comparación e *Index2ClassTable* una función de mapeo para mapear los resultados de la comparación con el índice de clase. La función de comparación $f(x,y)$ se define de la siguiente manera:

si $x-y >= th$,		$f(x,y) = 1$,
si $x=y$,	$f(x,y) =0$, y	
si $x-y<th$,	$f(x,y) =-1$	

en la que *th* es un umbral.

Una función de comparación alternativa $f(x, y)$ puede definirse de la siguiente manera:

si $(x/s)-(y/s)>=th$,	$f(x,y) = 1$,
si $(x/s)=(y/s)$,	$f(x,y) =0$, y
si $(x/s)-(y/s)<th$,	$f(x,y) =-1$.

en la que *th* es un umbral y *s* es un factor de escala.

El procedimiento de clasificación de EO mejorado puede ser aplicado al procedimiento de clasificación de píxeles combinado de SAO y otros procedimientos de clasificación de píxeles. Por ejemplo, una categoría de desplazamiento de borde (EO) según la clasificación de EO mejorada mencionada anteriormente puede ser dividida aún más por el EO o el BO.

En un enfoque convencional, la unidad utilizada para la codificación ALF basada en LCU es siempre una LCU. La mejora en la distorsión debida a ALF está relacionada con el tamaño de la LCU. Una LCU más pequeña generalmente permite que el diseño ALF se adapte más a las características locales. Sin embargo, la cantidad de información asociada con los parámetros de ALF es relativamente constante e independiente del tamaño de la LCU. Por lo tanto, un tamaño de LCU más pequeño resultará en una mayor tasa de bits asociada con la información correspondiente a los parámetros de ALF. Por consiguiente, puede reducirse sustancialmente la tasa de bits neta disponible para codificar datos de video y puede degradarse el rendimiento del sistema. Con el fin de superar este problema, una forma de realización de acuerdo con la presente invención agrupa varias LCU en una unidad, denominada unidad de filtro o bloque de codificación en esta divulgación. En consecuencia, se aplica el mismo ALF a todas las LCU en una unidad de filtro y los parámetros de ALF son compartidos entre todas las LCU en una unidad de filtro para reducir la tasa de bits necesaria para incorporar los parámetros de ALF. El conjunto de parámetros de ALF puede incluir uno o más elementos seleccionados de un conjunto que consiste en coeficientes de filtro, forma del filtro, tamaño del filtro, control de ON/OFF e información de región. Se puede usar un búfer para almacenar el conjunto de parámetros de ALF/SAO para que la información pueda ser compartida por otra unidad de filtro, segmento o imagen. La unidad de filtro puede ser tan grande como una imagen o varias LCU. Por ejemplo, una unidad de filtro puede consistir en MxN LCU, en el que M y N son números enteros mayores que cero. Los límites de una unidad de filtro pueden estar alineados o no con los límites de la LCU. Cuando se utiliza una unidad de filtro, los parámetros de ALF pueden ser diseñados en función de las estadísticas asociadas con la unidad de filtro. Los parámetros de ALF diseñados pueden aplicarse a todos los píxeles en la unidad de filtro. El codificador puede determinar el valor de M y N e incorporar la información del tamaño del filtro en una RBSP (Carga útil de secuencias de bytes sin procesar) de capas de secuencias o RBSP de capas de imágenes. Por lo tanto, se puede reducir la información complementaria correspondiente a ALF compartiendo los parámetros en bucle entre múltiples LCU.

Los candidatos a filtro para una unidad de filtro se pueden derivar en base a las unidades de filtro subyacentes, o los candidatos a filtro pueden compartir al menos una parte de los candidatos a filtro utilizados por las unidades de filtro

procesadas anteriormente en el segmento actual. Sin embargo, para la primera unidad de filtro en un segmento, no hay una unidad de filtro procesada anteriormente en el segmento para que la unidad de filtro actual la comparta. Por lo tanto, deben usarse candidatos a filtro predeterminados u otros medios para procesar la primera unidad de filtro y puede degradarse el rendimiento. En un enfoque convencional, los parámetros de ALF son derivados a partir de una unidad de filtro (una LCU o una imagen) sin ninguna información de otras unidades de filtro. Una forma de realización de acuerdo con la presente invención permite usar la información de un fotograma anterior o unidades de filtro procesadas anteriormente para derivar algunos candidatos a filtro para una unidad de filtro actual. Por ejemplo, se pueden usar las estadísticas de unidades de filtro con el modo ALF-OFF en el fotograma anterior para derivar parámetros para un filtro y se puede usar el filtro como un candidato a filtro para la unidad de filtro actual. El filtro derivado a partir de las unidades de filtro ALF-ON en el fotograma anterior puede ser utilizado como otro candidato a filtro. Además, una imagen se puede dividir en varias particiones y se pueden derivar los respectivos filtros para las unidades de filtro ALF-ON y ALF-OFF en cada partición. Uno de estos filtros se puede utilizar como un candidato a filtro para las unidades de filtro actuales. Las estadísticas de las unidades procesadas anteriormente en un segmento actual se pueden acumular para derivar candidatos a filtro en función de las estadísticas acumuladas.

De acuerdo con el procesamiento de ALF convencional, el procesamiento de componentes de croma puede ser independiente del procesamiento del componente de luma o siempre se realiza después del procesamiento del componente de luma. Una forma de realización de acuerdo con la presente invención combina ambos procedimientos de forma adaptativa. Se usa una sintaxis para indicar si el componente de croma es filtrado o no, y se usa otra sintaxis para indicar si los componentes de croma comparten el filtro con el componente de luma o utilizan su propia versión incorporada en el flujo de bits. Por lo tanto, los coeficientes de filtro para los componentes de croma se pueden derivar a partir del filtro luma o decodificar a partir del flujo de bits. Además, con el fin de reducir la información complementaria asociada con los parámetros de ALF, la huella de filtro para los componentes de croma puede ser un subconjunto de los filtros para el componente de luma.

Cuando se comparten parámetros de ALF anteriores, los parámetros en unidades de filtro anteriores del segmento actual pueden ser reutilizados por las unidades de filtro posteriores. Para el procesamiento de SAO, los parámetros en regiones o LCU anteriores del segmento actual pueden ser reutilizados por las siguientes regiones o LCU. Por conveniencia, la unidad de filtro también se puede referir a una región o una LCU. Lo más atrás que puede estar la posibilidad de reutilización de los parámetros de ALF anteriores, puede ser definido por un usuario o puede depender del tamaño de imagen. Cuando la información de ALF/SAO es compartida, la información de ALF/SAO se puede derivar o copiar de una región codificada anteriormente, una imagen codificada anteriormente, o información de ALF/SAO predefinida. La información de ALF puede incluir uno o más elementos del conjunto que consiste en coeficientes de filtro, forma de filtro, tamaño de filtro, control ON/OFF e información de región.

Cada unidad de filtro puede utilizar un índice para seleccionar parámetros de ALF/SAO anteriores almacenados en un búfer. El índice se puede codificar por entropía de forma adaptativa para reducir la tasa de bits. Por ejemplo, se puede asignar una palabra de código más corta a un conjunto de parámetros de ALF/SAO seleccionados con mayor frecuencia. En otro ejemplo, se puede usar una técnica de codificación predictiva en la que se determinan uno o más modos más probables. Si el índice actual es igual a uno de los modos más probables, se puede usar una palabra de código muy corta para codificar el índice. De lo contrario, se requerirán más bits para identificar cuál de los índices restantes es el mismo que el índice actual. La técnica es similar a la técnica de codificación de modo más probable utilizada en la codificación modo intra. La sintaxis correspondiente para el índice puede ser incorporada a un conjunto de parámetros de adaptación (APS), conjunto de parámetros de imagen (PPS), cabecera de segmento, o datos de segmento. La información asociada con ALF se puede incorporar al conjunto de parámetros de adaptación (APS), cabecera de segmento, datos de segmento, o se puede cambiar de forma adaptativa en función de un indicador incorporado en el conjunto de parámetros de secuencia (SPS), conjunto de parámetros de imagen (PPS), conjunto de parámetros de adaptación (APS), o una cabecera de segmento de acuerdo con una forma de realización de la presente invención.

En otro ejemplo para habilitar la señalización adaptativa de parámetros de ALF, se puede usar un indicador de intercalación de parámetros de ALF o un indicador de intercalación de parámetros de SAO para indicar si los parámetros de ALF para cada unidad de filtro (según se ha mencionado anteriormente, la unidad de filtro puede referirse a una región de una LCU) son intercalados con datos de unidad de filtro en el segmento. El indicador de intercalación de parámetros ALF o el indicador de intercalación de parámetros de SAO pueden ser incorporados en el conjunto de parámetros de imagen (PPS), conjunto de parámetros de adaptación (APS) o una cabecera de segmento. Además, el indicador para indicar el intercalado puede ser incorporado en múltiples RBSP simultáneamente, tal como un APS y una cabecera de segmento. Cuando el indicador para indicar el intercalado existe en múltiples RBSP, el indicador debe tener el mismo valor en los múltiples RBSP. Por consiguiente, la forma de realización de la presente invención también proporciona la capacidad de redundancia para proteger el indicador de intercalación.

Para el SAO, se puede usar un indicador, sao_max_region_size_minus_one, para indicar el número máximo de LCU en una región de procesamiento de SAO. El número máximo de LCU para una región de procesamiento de SAO es sao_max_region_size_minus_one + 1. El indicador, sao_max_region_size_minus_one puede ser incorporado al conjunto de parámetros de imagen (PPS), conjunto de parámetros de secuencia (SPS), conjunto de parámetros de adaptación (APS), cabecera de segmento o más de uno de los anteriores. Las LCU en la misma región de procesamiento de SAO pueden compartir los mismos parámetros de SAO. Por ejemplo, si el sao_max_region_size_minus_one es cero, el número máximo de LCU en una región de procesamiento de SAO es igual a uno. Si se utiliza un diseño de sintaxis basado en LCU y una codificación "run", el valor "run" indica el número de LCU que comparten los mismos parámetros de SAO. Para algunas aplicaciones, se puede usar un tamaño de LCU pequeño, tal como 16x16. En este caso, la tasa de bits asociada con la codificación run-length de LCU consecutivas en una región que comparte los parámetros de SAO puede ser relativamente alta. Por lo tanto, el valor "run" se usa para indicar el número de regiones que comparten los mismos parámetros de SAO.

La codificación run-length también se puede aplicar a los parámetros de ALF. Una unidad de filtro puede consistir en una LCU y el procesamiento de filtro basado en unidades pasa a ser un procesamiento basado en LCU en este caso. Unidades de filtro consecutivas pueden compartir los mismos parámetros de ALF. Para reducir la tasa de bits para indicar el uso compartido de parámetros de ALF, se utiliza la codificación run-length para indicar el número de unidades de filtro consecutivas que comparten los parámetros de ALF con la unidad de filtro actual.

Para el caso de segmento de granularidad fina, el bloque de granularidad puede ser menor que una LCU. En este caso, una LCU puede incluir más de un segmento de datos, es decir, una LCU puede contener más de un conjunto de parámetros de ALF/SAO. Una forma de realización de acuerdo con la presente invención obligará a que todos los parámetros del filtro en bucle en una LCU sean los mismos.

Un diseño en bucle basado en imagen debe esperar hasta que esté disponible la imagen completa. Esto puede provocar un retraso en el procesamiento. Sin embargo, puede que no sea un problema para las imágenes que no son de referencia. Una forma de realización de acuerdo con la presente invención puede simplemente omitir la etapa de filtrado real después de que se hayan derivado los parámetros del filtro. En cambio, en un enfoque basado en imagen, el procesamiento en bucle se puede aplicar a imágenes que no son de referencia sin provocar una latencia de codificación extra o un acceso adicional a la memoria de imágenes.

El procesamiento de filtro en bucle en el lado del codificador implica dos etapas separadas. En la primera etapa, se recopilan las estadísticas de la LCU o imagen subyacente. Entonces, se derivan parámetros de filtro en bucle según las estadísticas recopiladas. En la segunda etapa, se aplica el procesamiento de filtro en bucle a los píxeles en la LCU o la imagen en base a los parámetros de filtro en bucle derivados. Dado que cada procesamiento en bucle, tal como el procesamiento de ALF, se realiza en etapas separadas, puede provocar un acceso a datos considerable. Una forma de realización de acuerdo con la presente invención combina una de las etapas de procesamiento de filtro en bucle con otro procesamiento de filtro en bucle para reducir el acceso a datos asociados. Por ejemplo, la primera etapa del procesamiento de ALF/SAO es recopilar estadísticas, que se puede realizar junto con otra herramienta de codificación basada en LCU, tal como un proceso de desbloqueo. En consecuencia, las estadísticas para el procesamiento de ALF/SAO se pueden recopilar durante el proceso de desbloqueo. En consecuencia, los parámetros de ALF/SAO se pueden derivar sin ningún acceso adicional a la memoria de imágenes. La segunda etapa del procesamiento de ALF/SAO incluye aplicar el filtrado a los datos de píxeles, que se puede realizar durante el proceso posterior de estimación de movimiento. Por lo tanto, se puede realizar el proceso ALF/SAO en el lado del codificador sin ningún acceso a memoria por separado, lo que se denomina codificación de cero etapas en esta divulgación. En el lado del decodificador, no es necesario recopilar estadísticas para el diseño del filtro ALF/SAO. Sin embargo, el decodificador aún puede aprovechar la decodificación de cero etapas realizando el procesamiento de filtro ALF/SAO durante la compensación de movimiento.

Para algunas aplicaciones de latencia baja, se prefiere el procesamiento de filtro basado en unidades. Para la codificación de filtro basada en unidades, se espera que la decisión de control ON/OFF de una unidad de filtro finalice una vez que los resultados de la codificación de la unidad de filtro estén disponibles. Además, se prefiere que los datos comprimidos asociados con la unidad de filtro sean intercalados con parámetros del filtro en bucle en un segmento. Para aplicaciones de latencia baja, el control ON/OFF a nivel de segmento puede provocar una latencia de codificación alta. Por consiguiente, una forma de realización de la presente invención siempre establece explícitamente el indicador de control ON/OFF a nivel de segmento para indicar ON (activación) cuando los parámetros del filtro en bucle son codificados e intercalados con datos de la unidad de filtro (también denominados datos de bloque en esta divulgación) en un segmento. Alternativamente, una forma de realización de la presente invención puede deshabilitar condicionalmente el indicador de control ON/OFF a nivel de segmento. Si los parámetros del filtro en bucle son codificados e intercalados en un segmento, no se envía el indicador de control ON/OFF a nivel de segmento. De lo contrario, se envía el indicador de control ON/OFF a nivel de segmento.

En un enfoque convencional, el número de desplazamientos en cada región para el BO y el EO es diferente. Una forma de realización de acuerdo con la presente invención unifica el número de desplazamientos en cada región para el BO y el EO. En consecuencia, se cambia el número de grupos de BO a ocho, y cada grupo de BO tiene cuatro desplazamientos para cada región. Reducir el número de desplazamientos para cada grupo puede reducir la tasa de bits asociada. La tasa de bits asociada con los desplazamientos también se puede reducir de acuerdo con una forma de realización de la presente invención. Por ejemplo, se puede restringir el rango de desplazamientos a un rango más pequeño. Esto será útil para regiones pequeñas en las que se espera que los desplazamientos sean más pequeños. El búfer necesario para la predicción de desplazamientos basada en bloques vecinos puede ser reducido de acuerdo con una forma de realización de la presente invención. Por ejemplo, la predicción de desplazamientos puede evitar el uso del desplazamiento para la LCU por encima de la LCU actual. Las operaciones de filtrado en bucle incluyen filtro de desbloqueo, SAO, y ALF. El procesamiento de SAO de croma puede mejorarse de acuerdo con una forma de realización de la presente invención habilitando condicionalmente el procesamiento de SAO de croma dependiendo de si el procesamiento de SAO de luma está habilitado. En otra forma de realización de acuerdo con la presente invención, pueden compartirse los desplazamientos para los componentes de croma y de luma cuando se selecciona el EO.

En el presente documento se ilustran diversos diseños de sintaxis como ejemplos de formas de realización según la presente invención. La figura 4 ilustra una estructura de sintaxis de conjunto de parámetros de secuencia de acuerdo con la presente invención, en la que se incorpora un loop_filter_across_slice_flag. Si el loop_filter_across_slice_flag es igual a uno, las operaciones de filtrado en bucle se pueden realizar a través del límite del segmento. Si el loop_filter_across_slice_flag es igual a cero, las operaciones de filtrado en bucle son independientes del segmento y las operaciones no cruzarán los límites del segmento. Por consiguiente, puede reducirse el búfer necesario para la predicción de parámetros de filtro en bucle en base a bloques vecinos de acuerdo con una forma de realización de la presente invención.

La figura 5 ilustra un ejemplo de incorporación de un indicador de intercalación de parámetros de SAO en un APS. En la figura 5, aps_id identifica el conjunto de parámetros de adaptación (APS) que es referenciado por bloques de codificación en un segmento con el correspondiente aps_id en la cabecera del segmento. Cuando el indicador de intercalación de parámetros de SAO, es decir, aps_sao_interleaving_flag es igual a uno, los parámetros de SAO son intercalados con datos de la unidad de filtro para los segmentos que referencian al APS según indica el aps_id. Cuando el aps_sao_interleaving_flag es igual a cero, los parámetros de SAO son incorporados en el APS para los segmentos que referencian al APS. Otros indicadores, es decir, aps_sample_adaptive_offset_flag son incorporados en el APS para controlar el filtro ON/OFF. Si el aps_sample_adaptive_offset_flag es igual a uno, el SAO está ON (activado) para los segmentos que referencian al APS según lo indicado por el aps_id. Por otro lado, si el aps_sample_adaptive_offset_flag es igual a cero, el SAO está OFF (desactivado) para los segmentos que referencian al APS según lo indicado por el aps_id.

La figura 6 ilustra un diseño de sintaxis de ejemplo, aps_sao_paramo(), para parámetros de SAO en el APS de acuerdo con una forma de realización de la presente invención. La estructura de sintaxis incluye la información de SAO necesaria para el procesamiento de SAO. Por ejemplo, la estructura de sintaxis puede incluir un indicador o indicadores, tales como sao_cb_enable_flag y sao_cr_enable_flag en la figura 6 para determinar si el procesamiento de SAO se aplica a los respectivos componentes de croma de la imagen actual. La estructura de sintaxis también puede incluir información sobre el tamaño de imagen en términos de tamaño de unidad de codificación más grande, tal como sao_num_lcu_in_width_minus1 y sao_num_lcu_in_height_minus1 en la figura 6. La estructura de sintaxis también puede incluir información sobre si todas las unidades de codificación en un segmento son procesadas utilizando los mismos parámetros de SAO o parámetros de SAO individuales para cada unidad de codificación más grande o unidad de filtro, según lo indicado por sao_one_luma_unit_flag, sao_one_cb_unit_flag y sao_one_cr_unit_flag respectivamente en la figura 6. Si alguno de los indicadores anteriores tiene un valor igual a uno, los valores de desplazamiento de SAO serán incorporados en los respectivos sao_offset_vlc() para compartición por parte de los bloques de codificación del segmento. La estructura de sintaxis también puede incluir una indicación, tal como sao_repeat_row_flag[cldx], con respecto a si los parámetros de SAO de los bloques de codificación en la fila de bloque de codificación actual, tal como una fila de LCU, son los mismos que los del bloque de codificación superior para el respectivo índice de componente de color, cldx.

En la estructura de sintaxis de ejemplo anterior, aps_sao_param(), también se incorpora la estructura de bloque SAO, sao_unit_vlc(), junto con el respectivo indicador de repetición-fila. La figura 7 ilustra una estructura de sintaxis de ejemplo, sao_unit_vlc(), que incluye información asociada con la información de compartición de parámetros de SAO entre bloques de codificación, tal como sao_run_diff y sao_merge_up_flag. El número de veces que se repiten los parámetros de SAO correspondientes a un bloque de codificación para bloques de codificación subsiguientes en la misma fila es representado por saoRun[cldx][rx][ry]. El índice de matriz cIdx especifica el componente de color; cIdx tiene un valor igual a 0, 1 o 2 correspondiente a luma, Cb o Cr respectivamente. Los índices de matriz srx y ry especifican la ubicación del bloque de codificación subyacente en relación con el bloque de codificación superior izquierdo de la imagen. El elemento de sintaxis, sao_run_diff especifica el saoRun[ ][ ][ ] del bloque de codificación

actual si la fila actual es la primera fila; de lo contrario, especifica la diferencia entre el run del bloque de codificación actual y el run del bloque de codificación superior. Cuando saoRun[ ][ ][ ] es mayor que o igual a cero, los elementos de sintaxis en sao_offset_vlc() se derivan a partir de los correspondientes elementos de sintaxis del bloque de codificación izquierdo. La longitud del elemento de sintaxis sao_run_diff es igual a Ceil(Log2(sao_num_lcu_in_width_minus 1 - rx + 2)) bits. Cuando el indicador sao_merge_up_flag es igual a uno, los elementos de sintaxis en sao_offset_vlc() se derivan a partir de los correspondientes elementos de sintaxis del bloque de codificación superior. Cuando el indicador sao_merge_up_flag es igual a cero, los elementos de sintaxis en sao_offset_vlc() no se derivan a partir de los correspondientes elementos de sintaxis del bloque de codificación superior. Cuando sao_merge_up_flag no está presente, se infiere que es igual a cero.

La figura 8 ilustra una estructura de sintaxis de ejemplo para sao_offset_vlc() de acuerdo con una forma de realización de la presente invención. El elemento de sintaxis sao_type_idx[cldx][rx][ry] indica el tipo de SAO que puede ser BO (Band Offset = desplazamiento de banda) o EO (Edge Offset = desplazamiento de borde). Cuando sao_type_idx[cldx][rx][ry] tiene un valor igual a 0, indica que el SAO está OFF (desactivado); un valor igual a uno hasta cuatro, indica que se usa una de las cuatro categorías del EO correspondientes a 0°, 90°, 135° y 45°; y un valor igual a cinco, indica que se usa el BO. En el ejemplo anterior, los tipos BO y EO tienen cuatro valores de desplazamiento de SAO.

La figura 9 ilustra una estructura de sintaxis de ejemplo para una cabecera de segmento para permitir que los parámetros de SAO sean incorporados en una cabecera de segmento de forma adaptativa de acuerdo con una forma de realización de la presente invención. Cuando SAO está habilitado, entre otras condiciones, según lo indicado por el indicador, sample_adaptive_offset_enabled_flag, se incorporan dos indicadores adicionales, es decir, slice_sao_interleaving_flag y slice_sample_adaptive_offset_flag en la cabecera de segmento. Si slice_sao_interleaving_flag es igual a uno, los parámetros de SAO son intercalados con datos de la unidad de filtro en los datos del segmento. Si slice_sao_interleaving_flag es igual a cero, los parámetros de SAO usan información incorporada en el APS referenciado por el aps_id. El indicador para denotar el intercalado se puede incorporar en múltiples RBSP simultáneamente, tales como APS y cabecera de segmento. Cuando existe el indicador para indicar el intercalado en diversos RBSP, el indicador debe tener el mismo valor en los múltiples RBSP. En consecuencia, cuando hay un APS activo, el valor del slice_sao_interleaving_flag en la cabecera del segmento será el mismo que el del aps_sao_interleaving_flag en el APS. Si slice_sample_adaptive_offset_flag es igual a uno, el SAO está activado (ON) para el segmento actual. Si slice_sample_adaptive_offset_flag es igual a cero, el SAO está desactivado (OFF) para el segmento actual. De manera similar, cuando hay un APS activo, el valor del slice_sample_adaptive_offset_flag será el mismo que el del aps_sample_adaptive_offset_flag. Por consiguiente, la forma de realización de la presente invención también proporciona la capacidad de redundancia para proteger el indicador de intercalación.

La figura 10 ilustra una estructura de sintaxis de ejemplo, slice_data(), para que los datos del segmento admitan la señalización de parámetros de SAO adaptativa de acuerdo con una forma de realización de la presente invención. Según se muestra en la figura 10, cuando slice_sao_interleaving_flag tiene un valor igual a uno, los datos de una unidad de SAO individual (es decir, sao_unit_cabac()) son incorporados si el respectivo indicador de habilitación de SAO está activado (ON).

La figura 11 ilustra una estructura de sintaxis de ejemplo, sao_unit_cabac(), para datos de unidad de SAO de acuerdo con una forma de realización de la presente invención. Los indicadores de fusión, sao_merge_left_flag y sao_merge_up_flag se utilizan para indicar si el bloque de codificación actual comparte desplazamientos SAO con la unidad de codificación izquierda o superior respectivamente. Cuando la unidad de codificación actual no comparte parámetros de SAO con su bloque vecino izquierdo o superior, se incorporan los desplazamientos SAO, sao_offset_cabac(), para el bloque de codificación actual.

La figura 12 ilustra una estructura de sintaxis de ejemplo para sao_offset_cabac() de acuerdo con una forma de realización de la presente invención. El elemento de sintaxis sao_type_idx[cldx][rx][ry] indica el tipo de SAO que puede ser BO (Band Offset = desplazamiento de banda) o EO (Edge Offset = desplazamiento de borde). Cuando sao_type_idx[cldx][rx][ry] tiene un valor igual a 0, indica que el SAO está OFF (desactivado); un valor igual a uno hasta cuatro, indica que se usa uno de los cuatro OE correspondientes a 0°, 90°, 135° y 45°; y un valor igual a cinco, indica que se usa BO. En el ejemplo anterior, tanto el tipo BO como el tipo EO tienen cuatro valores de desplazamiento de SAO. Cuando el sao_type_idx[cldx][rx][ry] no está presente, se puede inferir el sao_type_idx[cldx][rx][ry]. Por ejemplo, si sao_merge_up_flag es igual a uno, el sao_type_idx[cldx][rx][ry] se establece igual al sao_type_idx[cldx][rx][ry-1]. De lo contrario, se establece el sao_type_idx[cldx][rx][ry] para que sea igual al sao_type_idx[cldx][rx-1][ry].

La figura 13 ilustra un ejemplo de diseño de sintaxis seq_parameter_set_rbsp(), para que el SPS admita un indicador para incorporar información de ALF de manera adaptativa. Según se muestra en la figura 13, el indicador, adaptive_loop_filter_enabled_flag se usa para indicar si se permite la incorporación adaptativa de parámetros de

ALF. Cuando la incorporación adaptativa de parámetros de ALF es permitida según lo indicado por el adaptive_loop_filter_enabled_flag, se usa otro indicador, alf_coef_in_slice_flag, para indicar dónde son incorporados los parámetros de ALF. Cuando alf_coef_in_slice_flag es igual a uno, la sintaxis alf_param() para parámetros de ALF es incorporada en la cabecera del segmento. Cuando alf_coef_in_slice_flag es igual a cero, la sintaxis alf_param() para parámetros de ALF será incorporada en el APS. En la sintaxis a nivel de segmento, si alf_coef_in_slice_flag es igual a uno, los parámetros de ALF pueden ser incorporados en una cabecera de segmento. Además, los parámetros de control ON/OFF de la unidad de control de ALF no serán incorporados a nivel de segmento.

La figura 14 ilustra un diseño de cabecera de segmento de ejemplo que permite que los parámetros de ALF sean incorporados en la cabecera del segmento de forma adaptativa según los indicadores adaptive_loop_filter_enabled_flag y alf_coef_in_slice_flag mencionados en la figura 13. Cuando ALF está habilitado según lo indicado por adaptive_loop_filter_enabled_flag, se usa otro indicador, slice_adaptive_loop_filter_flag para indicar si se aplica el ALF a nivel de segmento. Si se aplica el ALF a nivel de segmento y alf_coef_in_slice_flag indica que los parámetros de ALF son incorporados en la cabecera de segmento, la sintaxis alf_param() es incorporada en la cabecera del segmento. Por otro lado, si se aplica el ALF a nivel de segmento y alf_coef_in_slice_flag indica que los parámetros de ALF no son incorporados en la cabecera del segmento, la sintaxis alf_cu_control_param() se incorporará en la cabecera del segmento.

La figura 15 ilustra un diseño de sintaxis de ejemplo para parámetros de ALF de acuerdo con una forma de realización de la presente invención. La estructura de sintaxis contiene la información de ALF necesaria para el procesamiento de ALF. Por ejemplo, la estructura de sintaxis puede incluir un indicador o indicadores, tales como alf_cb_enable_flag y alf_cr_enable_flag en la figura 15 para determinar si el procesamiento de ALF es aplicado a componentes de croma respectivos de la imagen actual. La estructura de sintaxis también puede incluir información sobre el tamaño de la imagen en términos del tamaño de la unidad de codificación más grande, tal como alf_num_lcu_in_width_minus1 y alf_num_lcu_in_height_minus1 en la figura 15. La estructura de sintaxis también puede incluir información sobre si todas las unidades de codificación en un segmento son procesadas utilizando los mismos parámetros de ALF o parámetros de ALF individuales para cada unidad de codificación más grande o unidad de filtro, según lo indicado por alf_one_luma_unit_flag, alf_one_cb_unit_flag y alf_one_cr_unit_flag respectivamente en la figura 15. La estructura de sintaxis también puede incluir una indicación, tal como alf_repeat_row_flag[cIdx], con respecto a si los parámetros de ALF de los bloques de codificación en la fila del bloque de codificación actual son los mismos que los del bloque de codificación superior para el respectivo índice de componente de color, cIdx.

La figura 16 ilustra una estructura de sintaxis de ejemplo, alf_unit(), para bloques de codificación ALF de acuerdo con una forma de realización de la presente invención. La sintaxis alf_unit() incluye información asociada con información de compartición de parámetros de ALF entre bloques de codificación, tal como alf_run_diff y alf_merge_up_flag. El número de veces que los parámetros de ALF correspondientes a un bloque de codificación son repetidos para bloques de codificación posteriores en la misma fila es representado por alfRun[cIdx][rx][ry]. El índice de matriz cIdx especifica el componente de color; cIdx tiene un valor igual a 0, 1 o 2 que corresponde a luma, Cb o Cr respectivamente. Los índices de matriz srx y ry especifican la ubicación del bloque de codificación subyacente en relación con el bloque de codificación superior izquierdo de la imagen. El elemento de sintaxis, alf_run_diff especifica el alfRun[ ][ ][ ] del bloque de codificación actual si la fila actual es la primera fila; de lo contrario, especifica la diferencia entre el run del bloque de codificación actual y el run del bloque de codificación superior. Cuando alfRun[ ][ ][ ] es mayor que o igual a cero, los elementos de sintaxis en alf_info() se derivan a partir de los correspondientes elementos de sintaxis del bloque de codificación izquierdo. La longitud del elemento de sintaxis alf_run_diff es igual a Ceil(Log2(alf_num_lcu_in_width_minus1 - rx + 2)) bits. Cuando el indicador, alf_merge_up_flag es igual a uno, los elementos de sintaxis en alf_info() se derivan a partir de los correspondientes elementos de sintaxis del bloque de codificación superior. Cuando el indicador, alf_merge_up_flag es igual a cero, los elementos de sintaxis en alf_info() no se derivan a partir de los correspondientes elementos de sintaxis del bloque de codificación superior. Cuando alf_merge_up_flag no está presente, se infiere que es igual a cero.

La figura 17 ilustra una estructura de sintaxis de ejemplo para información de ALF, alf_info() de acuerdo con una forma de realización de la presente invención. La sintaxis alf_info() incluye información asociada con el filtro ALF. El diseño de sintaxis de ejemplo admite el uso de un búfer ALF, de modo que se puede usar un índice para seleccionar uno de entre múltiples filtros ALF almacenados en el búfer ALF. Por ejemplo, cuando alf_new_filter_set_flag es igual a uno, indica que el bloque de codificación actual utiliza un nuevo conjunto de filtros. De lo contrario, el bloque de codificación actual utiliza el conjunto de filtros almacenados con el índice del búfer igual a alf_stored_filter_set_idx[cIdx]. Cuando alf_new_filter_set_flag no está presente, se infiere que es igual a uno. Cuando alf_new_filter_set_flag es igual a uno, se incrementa NumALFFiltersInStoredBuffer[cIdx] en uno, en el que NumALFFiltersInStoredBuffer[cIdx] es el número de filtros en el conjunto de filtros. La sintaxis alf_stored_filter_set_idx[cIdx] especifica el índice de búfer de los filtros almacenados para el componente de color cIdx. La longitud del elemento de sintaxis alf_stored_filter_set_idx[cIdx] es igual a Floor(Log2(Min(1, NumALFFiltersInStoredBuffer[cIdx] - 1))) + 1 bits. El

elemento de sintaxis alf_no_filters_minus1 se usa para derivar el número de conjuntos de filtros para el bloque de codificación actual, en el que alf_no_filters_minus1 tiene un valor igual a entre 0 y 2. El elemento de sintaxis alf_start_second_filter especifica el índice de modo adaptativo de bloques (BA, block adaptive) de muestras de luma para las que se aplica el segundo filtro. Cuando el elemento de sintaxis alf_filter_pattern_flag[cldx][ry][rx][i] es igual a uno, se incrementa en uno el índice de filtro para el i-ésimo índice de modo BA. Cuando el elemento de sintaxis alf_filter_pattern_flag[cldx][ry][rx][i] es igual a cero, el índice de filtro para el i-ésimo índice de modo BA es el mismo que el (i-1)-ésimo índice de modo BA. Cuando el elemento de sintaxis alf_pred_flag[ ][ ][ ][ ] es igual a uno, los coeficientes de filtro para el bloque de codificación actual son codificados de manera predictiva; de lo contrario los coeficientes de filtro son codificados directamente. El elemento de sintaxis alf_min_kstart_minus1 + 1 especifica el orden k mínimo del k-ésimo orden exponencial del código de Golomb para los coeficientes de filtro de luma para el filtro en bucle adaptativo. El elemento de sintaxis alf_golomb_index_flag[i] especifica la diferencia en el orden del k-ésimo orden exponencial de códigos de Golomb entre el i-ésimo grupo y el (i+1)-ésimo grupo de los coeficientes de filtro de luma. Existen múltiples grupos de los coeficientes de filtro de luma en los que cada grupo puede tener un orden k diferente. Cuando el elemento de sintaxis alf_nb_pred_luma_flag[cldx][ry][rx][i] es igual a uno, los coeficientes de filtro del i-ésimo filtro para el bloque de codificación actual son codificados de forma predictiva en base a coeficientes de filtro espacialmente vecinos; de lo contrario, los coeficientes de filtro no son codificados utilizando coeficientes de filtro espacialmente vecinos. El elemento de sintaxis alf_filt_coeff[cldx][ry][rx][i] especifica el j-ésimo coeficiente de filtro del i-ésimo filtro usado en el proceso de filtrado en bucle adaptativo para el bloque de codificación actual.

Una forma de realización de sistemas de codificación de video que incorporan un procesamiento de SAO y/o ALF mejorado de acuerdo con la presente invención según se ha descrito anteriormente puede ser implementada en una diversidad de hardware, códigos de software o una combinación de ambos. Por ejemplo, una forma de realización de la presente invención puede ser un circuito integrado en un chip de compresión de video o códigos de programa integrados en un software de compresión de video para realizar el procesamiento descrito en el presente documento. Una forma de realización de la presente invención también puede ser códigos de programa a ejecutar en un Procesador de Señal Digital (DSP) para realizar el procesamiento descrito en el presente documento. La invención también puede implicar una serie de funciones a realizar por un procesador informático, un procesador de señal digital, un microprocesador o una matriz de puerta programable de campo (FPGA). Estos procesadores pueden estar configurados para realizar tareas particulares de acuerdo con la invención, ejecutando un código de software legible informáticamente o un código de firmware que define los procedimientos particulares incorporados en la invención. El código de software o los códigos de firmware pueden desarrollarse en diferentes lenguajes de programación y diferentes formatos o estilos. El código de software también se puede compilar para diferentes plataformas de destino. Sin embargo, diferentes formatos de código, estilos y lenguajes de códigos de software y otros medios de configuración de código para realizar las tareas de acuerdo con la invención no se apartarán del espíritu y alcance de la invención.

La invención puede ser realizada en otras formas específicas. Los ejemplos descritos deben considerarse solo como ilustrativos y no restrictivos en todos los aspectos. Por lo tanto, el alcance de la invención es indicado por las reivindicaciones adjuntas en lugar de por la descripción anterior. Todos los cambios que se encuentren dentro del significado y rango de equivalencia de las reivindicaciones deben incluirse dentro de su alcance.

En aras de la exhaustividad, la presente invención se describe en base a las siguientes cláusulas:

1. Un procedimiento para decodificar video con un procesamiento en bucle de un video reconstruido, comprendiendo el procedimiento:
recuperar unos datos de video reconstruido a partir de un flujo de bits de video;
recibir un indicador procedente del flujo de bits de video;
de acuerdo con el indicador, recibir información asociada con parámetros de un filtro en bucle procedente de una carga útil de datos en el flujo de bits de video a compartir por dos o más bloques de codificación o procedente de datos de bloques de codificación individuales en el flujo de bits de video; y
aplicar el procesamiento en bucle a los bloques de codificación del video reconstruido.

2. El procedimiento de la cláusula 1, en el que el procesamiento en bucle corresponde a un filtro en bucle adaptativo (ALF) o a un desplazamiento adaptativo de muestras (SAO).

3. El procedimiento de la cláusula 1, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples unidades de codificación, una unidad de codificación más grande (LCU) o múltiples LCU.

4. El procedimiento de la cláusula 1, en el que la carga útil de datos en el flujo de bits de video está en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

5. El procedimiento de la cláusula 1, en el que el indicador es un indicador de intercalación, en el que el indicador de intercalación se refiere a si la información asociada con los parámetros del filtro en bucle es incorporada en la carga útil de datos en el flujo de bits de video o intercalada con datos de bloques de codificación individuales en el flujo de datos de video.

6. El procedimiento de la reivindicación 5, en el que existen dos indicadores de intercalación en al menos dos cargas útiles de datos que corresponden a al menos dos miembros diferentes seleccionados de un grupo que consiste en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

7. El procedimiento de la cláusula 6, en el que los dos indicadores de intercalación tienen un mismo valor.

8. El procedimiento de la cláusula 6, en el que los dos indicadores de intercalación existen en el APS y la cabecera de segmento; en el que los dos indicadores de intercalación tienen un valor que indica que la información asociada con los parámetros del filtro en bucle es incorporada en el APS; y cada segmento incluye un identificador de APS que referencia al APS para permitir que los bloques de codificación en el segmento utilicen la información asociada con los parámetros del filtro en bucle incorporada en el APS.

9. El procedimiento de la cláusula 1, en el que el indicador está en un nivel de secuencia y en el que la información asociada con los parámetros del filtro en bucle es incorporada en el conjunto de parámetros de adaptación (APS), o una cabecera de segmento de acuerdo con el indicador.

10. El procedimiento de la cláusula 9, en el que el procesamiento en bucle corresponde a un filtro en bucle adaptativo (ALF); la cabecera de segmento comprende información asociada con los parámetros de ALF si el ALF está habilitado y el indicador indica que la información asociada con los parámetros de ALF está en la cabecera del segmento; y la cabecera del segmento comprende un parámetro de control de unidad de codificación si el ALF está habilitado y el indicador indica que la información asociada con los parámetros de ALF está en el APS.

11. Un procedimiento para codificar video con un procesamiento en bucle de un video reconstruido, comprendiendo el procedimiento:

recibir datos de video reconstruido;

determinar parámetros de un filtro en bucle asociados con el procesamiento en bucle, en el que el procesamiento en bucle es aplicado a bloques de codificación del video reconstruido; e

incorporar información asociada con los parámetros del filtro en bucle, o bien en una carga útil de datos en un flujo de bits de video a compartir por dos o más bloques de codificación, o bien intercalada con datos de bloques de codificación individuales en el flujo de bits de video de acuerdo con un indicador.

12. El procedimiento de la cláusula 11, en el que el procesamiento en bucle corresponde a un filtro en bucle adaptativo (ALF) o a un desplazamiento adaptativo de muestras (SAO).

13. El procedimiento de la cláusula 11, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples unidades de codificación, una unidad de codificación más grande (LCU) o múltiples LCU.

14. El procedimiento de la cláusula 11, en el que la carga útil de datos en el flujo de bits de video está en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

15. El procedimiento de la cláusula 11, en el que existen dos indicadores en al menos dos cargas útiles de datos que corresponden a al menos dos miembros diferentes seleccionados de un grupo que consiste en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

16. El procedimiento de la cláusula 15, en el que los dos indicadores tienen un mismo valor.

17. El procedimiento de la cláusula 15, en el que los dos indicadores existen en el APS y la cabecera de segmento; en el que los dos indicadores tienen un valor que indica que la información asociada con los parámetros del filtro en bucle es incorporada en el APS; y cada segmento incluye un identificador de APS que referencia al APS para permitir que los bloques de codificación en el segmento utilicen la información asociada con los parámetros del filtro en bucle incorporados en el APS.

18. El procedimiento de la cláusula 11, en el que el indicador está en un nivel de secuencia y en el que la información asociada con los parámetros del filtro en bucle es incorporada en el conjunto de parámetros de adaptación (APS), o una cabecera de segmento de acuerdo con el indicador.

19. El procedimiento de la cláusula 18, en el que el procesamiento en bucle corresponde a un filtro en bucle adaptativo (ALF); la cabecera del segmento incluye información asociada con los parámetros de ALF si el ALF está

habilitado y el indicador indica que la información asociada con los parámetros de ALF está en la cabecera del segmento; y la cabecera del segmento incluye un parámetro de control de unidad de codificación si el ALF está habilitado y el indicador indica que la información asociada con los parámetros de ALF está en el APS.

20. Un aparato para decodificar video con un procesamiento en bucle de un video reconstruido, que comprende:
medios para recuperar unos datos de video reconstruido a partir de un flujo de bits de video;
medios para recibir un indicador procedente del flujo de bits de video;
medios para recibir información asociada con parámetros de un filtro en bucle procedente de o bien una carga útil de datos en el flujo de bits de video a compartir por dos o más bloques de codificación, o bien procedente de datos de bloques de codificación individuales en el flujo de bits de video de acuerdo con el indicador; y
medios para aplicar el procesamiento en bucle a los bloques de codificación del video reconstruido.

21. Un aparato para codificar video con un procesamiento en bucle de un video reconstruido, que comprende:
medios para recibir datos de video reconstruido;
medios para determinar parámetros de un filtro en bucle asociados con el procesamiento en bucle, en el que el procesamiento en bucle es aplicado a bloques de codificación del video reconstruido; y
medios para incorporar información asociada con los parámetros del filtro en bucle, o bien en una carga útil de datos en un flujo de bits de video a compartir por dos o más bloques de codificación, o bien intercalada con datos de bloques de codificación individuales en el flujo de bits de video de acuerdo con un indicador.

22. Un procedimiento para decodificar video con un procesamiento de desplazamiento adaptativo de muestras (SAO) de un video reconstruido, comprendiendo el procedimiento:
recuperar unos datos de video reconstruido a partir de un flujo de bits de video;
recibir información asociada con parámetros de SAO procedente de unos datos de bloques de codificación individuales en el flujo de bits de video; y
aplicar el procesamiento de SAO a los bloques de codificación del video reconstruido;
en el que la información asociada con los parámetros de SAO está intercalada con datos de bloques de codificación individuales en el flujo de bits de video.

23. El procedimiento de la cláusula 22, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples unidades de codificación, una unidad de codificación más grande (LCU) o múltiples LCU.

24. Un procedimiento para codificar video con un procesamiento de desplazamiento adaptativo de muestras (SAO) de un video reconstruido, comprendiendo el procedimiento:
recibir datos de video reconstruido;
determinar unos parámetros de SAO asociados con el procesamiento de SAO, en el que el procesamiento de SAO es aplicado a bloques de codificación del video reconstruido; e
incorporar información asociada con los parámetros de SAO intercalándola con datos de bloques de codificación individuales en un flujo de bits de video.

25. El procedimiento de la cláusula 24, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples unidades de codificación, una unidad de codificación más grande (LCU) o múltiples LCU.

REIVINDICACIONES

1. Un procedimiento para decodificar video con un procesamiento de desplazamiento adaptativo de muestras, también denominado SAO en lo sucesivo, de un video reconstruido, comprendiendo el procedimiento:
recuperar unos datos de video reconstruido a partir de un flujo de bits de video;
recibir uno o más parámetros de SAO procedentes de unos datos de bloques de codificación individuales en el flujo de bits de video; y
aplicar el procesamiento de SAO a los bloques de codificación de los datos de video reconstruido;
caracterizado porque dicho uno o más parámetros de SAO son compartidos entre al menos dos componentes de color diferentes, y/o uno o más parámetros de SAO de un primer componente de color son derivados a partir de uno o más parámetros de SAO de un segundo componente de color.

2. El procedimiento de la reivindicación 1, en el que dicho uno o más parámetros de SAO comprenden un indicador de fusión, un indicador de fusión izquierda, un indicador de fusión superior, un tipo de SAO, un desplazamiento de SAO, o una combinación de los mismos.

3. El procedimiento de la reivindicación 2, en el que el tipo de SAO comprende una combinación de habilitación del procesamiento de SAO, deshabilitación del procesamiento de SAO, desplazamiento de banda y desplazamiento de borde.

4. El procedimiento de la reivindicación 1, en el que los componentes de color comprenden componente de color de luma, componente de color de croma Cb, y componente de color de croma Cr.

5. Un procedimiento para codificar video con un procesamiento de desplazamiento adaptativo de muestras, también denominado SAO en lo sucesivo, de un video reconstruido, comprendiendo el procedimiento:
recibir unos datos de video reconstruido; y
determinar uno o más parámetros de SAO para el procesamiento de SAO, en el que el procesamiento de SAO es aplicado a bloques de codificación de los datos de video reconstruido;
caracterizado por incorporar uno o más parámetros de SAO en un flujo de bits de video, en el que dicho uno o más parámetros de SAO son compartidos entre al menos dos componentes de color diferentes, y/o uno o más parámetros de SAO de un primer componente de color son derivados a partir de uno o más parámetros de SAO de un segundo componente de color.

6. El procedimiento de la reivindicación 5, en el que dicho uno o más parámetros de SAO comprenden un indicador de fusión, un indicador de fusión izquierda, un indicador de fusión superior, un tipo de SAO, un desplazamiento de SAO, o una combinación de los mismos.

7. El procedimiento de la reivindicación 5, en el que el tipo de SAO comprende uno o una combinación de habilitación del procesamiento de SAO, deshabilitación del procesamiento de SAO, desplazamiento de banda, y desplazamiento de borde.

8. El procedimiento de la reivindicación 5, en el que los componentes de color comprenden componente de color de luma, componente de color de croma Cb, componente de color de croma Cr.

9. Flujo de bits de vídeo que tiene codificada en el mismo
información codificada de datos de video reconstruido, y
uno o más elementos de sintaxis asociados con uno o más parámetros de SAO para un procesamiento de SAO, **caracterizado porque** dicho uno o más parámetros de SAO son compartidos entre al menos dos componentes de color diferentes, y/o uno o más parámetros de SAO de un primer componente de color son derivados a partir de uno o más parámetros de SAO de un segundo componente de color.

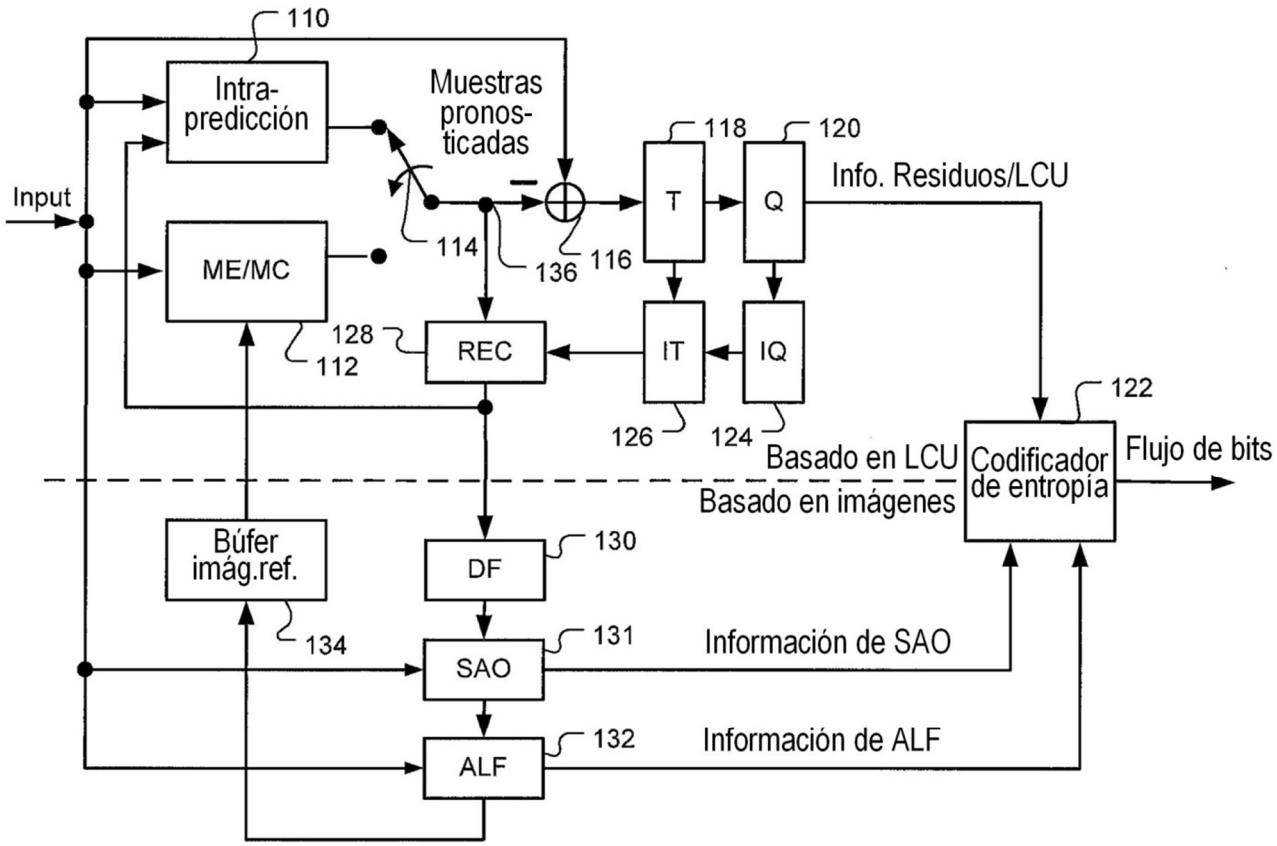
110
Intra-predicción
Muestras pronos-ticadas
118
120
Info. Residuos/LCU
Input
ME/MC
114
136
116
T
Q
128
112
REC
IT
IQ
126
124
122
Basado en LCU
Codificador de entropía
Flujo de bits
Basado en imágenes
Búfer imág.ref.
DF
130
134
131
Información de SAO
SAO
132
Información de ALF
ALF

Fig. 1A

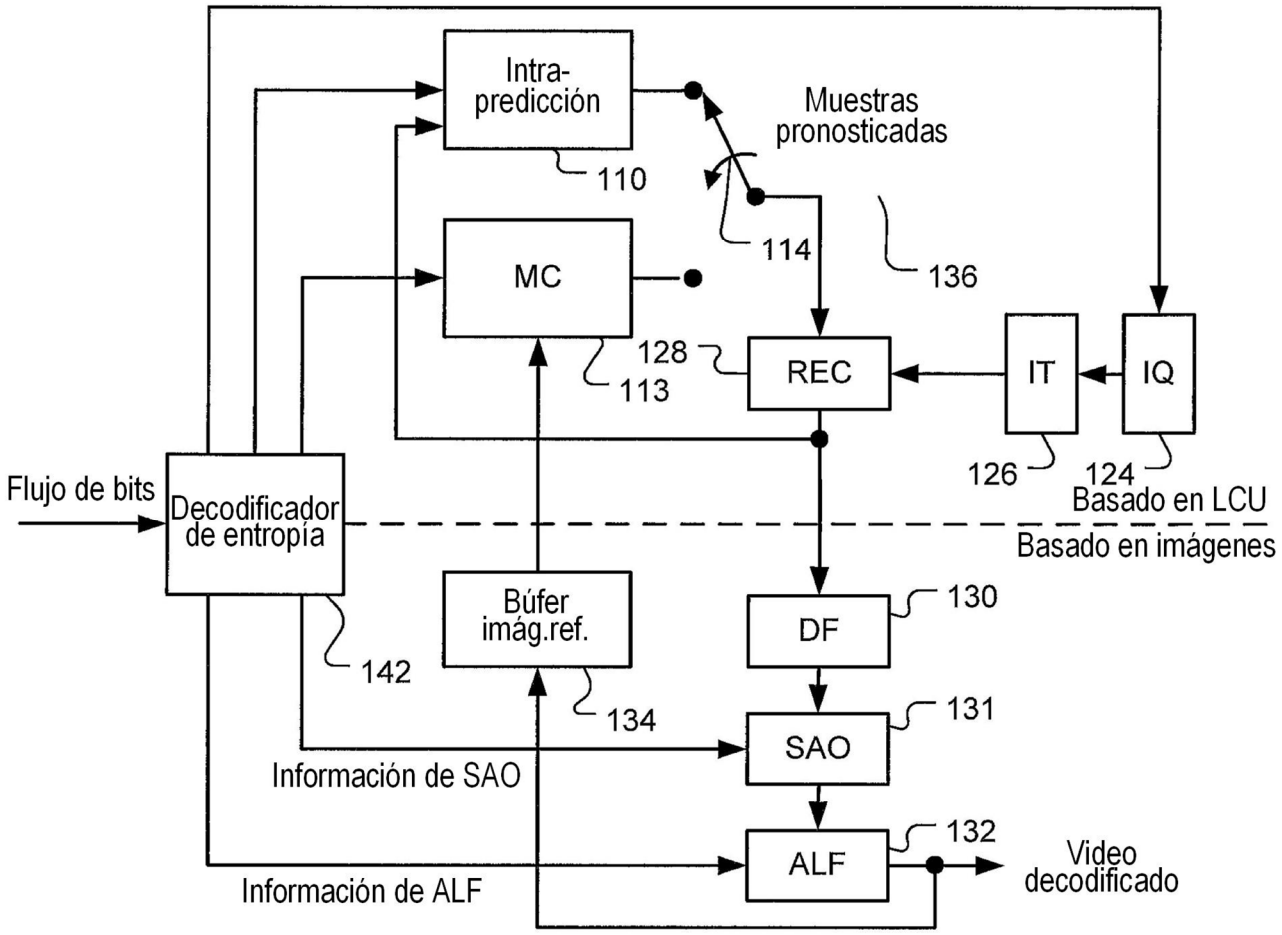
Intra-
predicción
110
Muestras
pronosticadas
114
136
MC
128
113
REC
IT
IQ
126
124
Basado en LCU
Basado en imágenes
Flujo de bits
Decodificador
de entropía
142
Búfer
imág.ref.
134
DF
130
SAO
131
Información de SAO
ALF
132
Video
decodificado
Información de ALF

Fig. 1B

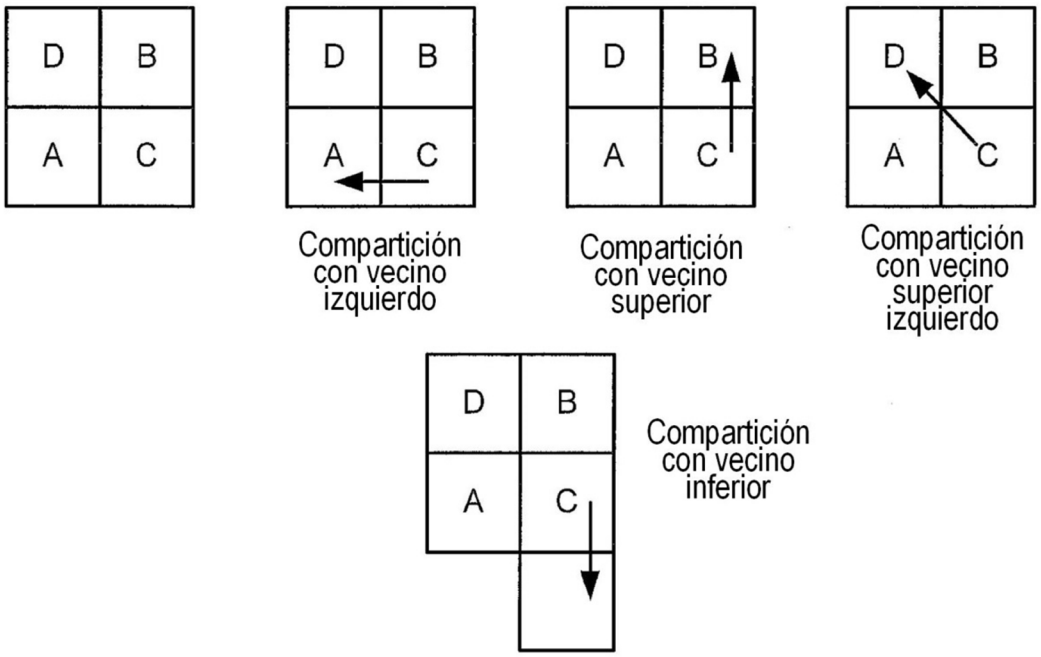
D
B
A
C
Compartición con vecino izquierdo
Compartición con vecino superior
Compartición con vecino superior izquierdo
Compartición con vecino inferior

Fig. 2

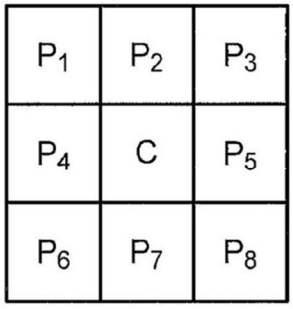
P1
P2
P3
P4
C
P5
P6
P7
P8

Fig. 3

seq_parameter_set_rbsp( ) {	**Descriptor**
…	.
loop_filter_across_slice_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
…	
}	

Fig. 4

aps_rbsp( ) {	**Descriptor**
aps_id	ue(v)
…	
aps_sao_interleaving_flag	u(1)
if(!aps_sao_interleaving_flag) {	
aps_sample_adaptive_offset_flag	u(1)
if(aps_sample_adaptive_offset_flag)	
aps_sao_param( )	
}	
…	
}	

Fig. 5

aps_sao_param( ) {	**Descriptor**
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
sao_num_lcu_in_width_minus1	ue(v)
sao_one_luma_unit_flag	u(1)
if (sao_one_luma_unit_flag)	
sao_offset_vlc(0, 0, 0)	
if (sao_cb_enable_flag){	
sao_one_cb_unit_flag	u(1)
if (sao_one_cb_unit_flag)	
sao_offset_vlc(0, 0, 1)	
}	
if (sao_cr_enable_flag){	
sao_one_cr_unit_flag	u(1)
if (sao_one_cr_unit_flag)	
sao_offset_vlc(0, 0, 2)	
}	
for (ry = 0; ry < sao_num_lcu_in_height_minus1+1; ry++) {	
for (rx = 0; rx < sao_num_lcu_in_width_minus1+1; rx++) {	
if (aps_sample_adaptive_offset_flag && ! sao_one_luma_unit_flag) {	
if (ry > 0 && rx == 0)	
sao_repeat_row_flag[0]	u(1)
sao_unit_vlc(rx, ry, 0)	
}	
if (sao_cb_enable_flag && ! sao_one_cb_unit_flag) {	
if (ry > 0 && rx == 0)	
sao_repeat_row_flag[1]	u(1)
sao_unit_vlc(rx, ry, 1)	
}	
if (sao_cr_enable_flag && !sao_one_cr_unit_flag) {	
if (ry > 0 && rx == 0)	
sao_repeat_row_flag[2]	u(1)
sao_unit_vlc(rx, ry, 2)	
}	
}	
}	
}	

Fig. 6

sao_unit_vlc(rx, ry, cIdx){	**Descriptor**
if (!sao_repeat_row_flag[cIdx]) {	
if (rx == 0 || run[cIdx][rx][ry] < 0) {	
if (ry == 0)	
saoRun[cIdx][rx][ry] = **sao_run_diff**	u(v)
else	
saoRun[cIdx][rx][ry] = **sao_run_diff** + saoRun[cIdx][rx][ry − 1]	se(v)
}	
saoRun[cIdx][rx + 1][ry] = saoRun[cIdx][rx][ry] − 1	
if (rx == 0 || saoRun[cIdx][rx][ry] < 0) {	
if (ry > 0)	
sao_merge_up_flag	u(1)
if (!sao_merge_up_flag)	
sao_offset_vlc(rx, ry, cIdx)	
}	
} else {	
saoRun[cIdx][ry][rx] = saoRun[cIdx][ry − 1][rx]	
}	
}	

Fig. 7

sao_offset_vlc(rx, ry, cIdx) {	**Descriptor**
sao_type_idx[cIdx][rx][ry]	ue(v)
if(sao_type_idx[cIdx][rx][ry] ==5) {	
sao_band_position[cIdx][rx][ry]	u(5)
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	se(v)
}	
else if(sao_type_idx[cIdx][rx][ry] != 0)	
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	ue(v)
}	

Fig. 8

slice_header( ) {	**Descriptor**
...	
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_sao_interleaving_flag	u(1)
slice_sample_adaptive_offset_flag	u(1)
if(slice_sao_interleaving_flag) {	
if(slice_sample_adaptive_offset_flag) {	
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
}	
}	
}	
aps_id	ue(v)
}	
...	
}	

Fig. 9

slice_data( ) {	Descriptor
…	
do {	
xCtb = horizontal location of largest coding unit	
yCtb = vertical location largest coding unit	
…	
CtbAddrInSlice = first largest coding unit location in current slice	
If above largest coding unit is available set AddrUp larger than 0, otherwise, set AddrUp = -1	
if(slice_sao_interleaving_flag) {	
if(slice_sample_adaptive_offset_flag)	
sao_unit_cabac(xCtb, yCtb, 0)	
if(sao_cb_enable_flag)	
sao_unit_cabac(xCtb, yCtb, 1)	
if(sao_cr_enable_flag)	
sao_unit_cabac(xCtb, yCtb, 2)	
}	
…	
} while(next largest coding unit data is exist)	
}	

Fig. 10

sao_unit_cabac(rx, ry, cIdx){	Descriptor
if (rx > 0 && CtbAddrInSlice != 0)	
sao_merge_left_flag	ae(v)
if (!sao_merge_left_flag) {	
if (ry > 0 && (AddrUp > 0 || loop_filter_across_slice_flag))	
sao_merge_up_flag	ae(v)
if (!sao_merge_up_flag)	
sao_offset_cabac(rx, ry, cIdx)	
}	
}	

Fig. 11

sao_offset_cabac(rx, ry, cIdx) {	**Descriptor**
sao_type_idx[cIdx][rx][ry]	ae(v)
if(sao_type_idx[cIdx][rx][ry] = =5)	
sao_band_position[cIdx][rx][ry]	ae(v)
if(sao_type_idx[cIdx][rx][ry] != 0)	
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	ae(v)
}	

Fig. 12

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)
.....	
adaptive_loop_filter_enabled_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	
alf_coef_in_slice_flag	u(1)
.....	
}	

Fig. 13

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)
…..	
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
……..	
}	

Fig. 14

alf_param() {	**Descriptor**
alf_cb_enable_flag	u(1)
alf_cr_enable_flag	u(1)
alf_one_luma_unit_per_slice_flag	
if(alf_cb_enable_flag)	
alf_one_cb_unit_per_slice_flag	u(1)
if(alf_cr_enable_flag)	
alf_one_cr_unit_per_slice_flag	
if(!alf_coef_in_slice_flag) {	
alf_num_lcu_in_width_minus1	ue(v)
alf_num_lcu_in_height_minus1	ue(v)
} else {	
alf_num_lcu_in_slice_minus1	u(v)
}	
endCtbrY = (numCtb – 1 + firstCtbAddr) / numCtbInWidth	
endCtbrX = (numCtb – 1 + firstCtbAddr) % numCtbInWidth	
for (i = 0; i < numCtb; i++) {	
rx = (i + firstCtbAddr) % numCtbInWidth	
ry = (i + firstCtbAddr) / numCtbInWidth	
endrX = (ry = = endCtbrY) ? (endCtbrX) : (numCtbInWidth – 1)	
if((rx = = 0) && (i – numCtbInWidth >= 0) && (alf_one_luma_unit_per_slice_flag = =0))	
alf_repeat_row _flag[0]	u(1)
alf_unit (rx, ry, 0, i, endrX, alf_one_luma_unit_per_slice_flag)	
if(alf_cb_enable_flag) {	
if((rx = = 0) && (i – numCtbInWidth >= 0) && (alf_one_cb_unit_per_slice_flag = =0))	
alf_repeat_row _flag[1]	u(1)
alf_unit (rx, ry, 1, i, endrX, alf_one_cb_unit_per_slice_flag)	
}	
if(alf_cr_enable_flag) {	
if((rx = = 0) && (i – numCtbInWidth>= 0) && (alf_one_cr_unit_per_slice_flag = =0))	
alf_repeat_row _flag[2]	u(1)
alf_unit (rx, ry, 2, i, endrX, alf_one_cr_unit_per_slice_flag)	
}	
}	
}	

Fig. 15

alf_unit(rx, ry, cIdx, lcuIdx, endrX, oneUnitFlag) {	**Descriptor**
if(oneUnitFlag) {	
if (lcuIdx = = 0) {	
alf_lcu_enable_flag[cIdx][ry][rx]	u(1)
if(alf_lcu_enable_flag[cIdx][ry][rx])	
alf_info(rx, ry, cIdx)	
}	
} else {	
if(!alf_repeat_row _flag[cIdx]) {	
if(rx = = 0 || alfRun[cIdx][ry][rx] < 0) {	
if(lcuIdx –numCtbInWidth < 0)	
alfRun[cIdx][ry][rx] = 0 + **alf_run_diff**	u(v)
else	
alfRun[cIdx][ry][rx] = alfRun[cIdx][ry – 1][rx]+ **alf_run_diff**	s(v)
}	
alfRun[cIdx][ry][rx+ 1] = alfRun[cIdx][ry][rx] – 1	
if (rx == 0 || alfRun [cIdx] [rx][ry] < 0) {	
if (ry > 0 && (lcuIdx – numCtbInWidth > = 0 | | alfAcrossSlice))	
alf_merge_up_flag	u(1)
if(!alf_merge_up_flag) {	
alf_lcu_enable_flag[cIdx][ry][rx]	u(1)
if(alf_lcu_enable_flag[cIdx][ry][rx])	
alf_info (rx, ry, cIdx)	
}	
}	
} else	
alfRun[cIdx][ry][rx] = alfRun[cIdx][ry – 1][rx]	
}	
}	

Fig. 16

alf_info (rx, ry, cIdx) {	**Descriptor**
if(NumALFFiltersInStoredBuffer[cIdx] > 0)	
alf_new_filter_set_flag	u(1)
if(alf_new_filter_set_flag = = 0 && NumALFFiltersInStoredBuffer[cIdx] >	
alf_stored_filter_set_idx[cIdx]	u(v)
} else {	
if(cIdx = = 0) {	
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 < 15; i++)	
alf_filter_pattern_flag[cIdx][ry][rx][i]	u(1)
}	
if(alf_no_filters_minus1 > 0)	
alf_pred_flag[cIdx][ry][rx]	u(1)
for (i = 0; i < AlfNumFilters; i++)	
alf_nb_pred_luma_flag[cIdx][ry][rx][i]	u(1)
if(AlfNumFilters > 1) {	
alf_min_kstart_minus1	ue(v)
for (i = 1; i < 4; i++)	
alf_golomb_index_flag[i]	u(1)
}	
for (i = 0; i < AlfNumFilters; i++) {	
for (j = 0; j < AlfCodedLength; j++)	
alf_filt_coeff[cIdx][ry][rx][i][j]	ge(v)
}	
} else {	
for (j = 0; j < AlfCodedLength; j++)	
alf_filt_coeff[cIdx][ry][rx][0][j]	se(v)
}	
}	
}	

Fig. 17