(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 748 100**

(51) Int. CI.:

H04N 19/176 (2014.01)
H04N 19/61 (2014.01)
H04N 19/82 (2014.01)
H04N 19/11 (2014.01)
H04N 19/593 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **14.07.2011 E 17170581 (7)**

(97) Fecha y número de publicación de la concesión europea: **04.09.2019 EP 3226560**

(54) Título: **Intra-predicción de baja complejidad para codificación de vídeo**

(30) Prioridad:

14.07.2010 US 364322 P
30.09.2010 US 388541 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
13.03.2020

(73) Titular/es:

NTT DOCOMO, INC. (100.0%)
11-1, Nagatacho 2-chome, Chiyoda-ku
Tokyo 100-6150, JP

(72) Inventor/es:

BOSSEN, FRANK JAN y
TAN, THIOW KENG

(74) Agente/Representante:

MARTÍN BADAJOZ, Irene

DESCRIPCIÓN

Intra-predicción de baja complejidad para codificación de vídeo

Antecedentes de la invención

1. Texto sobre el campo técnico

La presente invención se refiere a la codificación de vídeo y en particular a la predicción intra-trama en la que se predice un bloque de muestra, usando píxeles anteriormente codificados y reconstruidos de la misma trama de vídeo.

2. Información sobre antecedentes

El vídeo digital requiere una gran cantidad de datos para representar todas y cada una de las tramas de una secuencia de vídeo digital (por ejemplo, serie de tramas) de una manera sin comprimir. Para la mayoría de las aplicaciones no resulta viable transmitir vídeo digital sin comprimir a través de redes informáticas debido a limitaciones del ancho de banda. Además, el vídeo digital sin comprimir requiere una gran cantidad de espacio de almacenamiento. El vídeo digital se codifica normalmente de alguna manera para reducir los requisitos de almacenamiento y reducir los requisitos de ancho de banda.

Una técnica para codificar vídeo digital es la predicción inter-trama, o inter-predicción. La inter-predicción aprovecha redundancias temporales entre diferentes tramas. Las tramas de vídeo temporalmente adyacentes incluyen normalmente bloques de píxeles, que permanecen sustancialmente iguales. Durante el procedimiento de codificación, un vector de movimiento interrelaciona el movimiento de un bloque de píxeles en una trama con un bloque de píxeles similares en otra trama. Por consiguiente, no se requiere que el sistema codifique el bloque de píxeles dos veces, sino que en vez de eso codifica el bloque de píxeles una vez y proporciona un vector de movimiento para predecir el otro bloque de píxeles.

Otra técnica para codificar vídeo digital es la predicción intra-trama o intra-predicción. La intra-predicción codifica una trama o una parte de la misma sin referencia a píxeles en otras tramas. La intra-predicción aprovecha redundancias espaciales entre bloques de píxeles dentro de una trama. Dado que bloques de píxeles espacialmente adyacentes tienen generalmente atributos similares, la eficacia del procedimiento de codificación se mejora haciendo referencia a la correlación espacial entre bloques adyacentes. Esta correlación puede aprovecharse mediante predicción de un bloque objetivo basándose en modos de predicción usados en bloques adyacentes.

Sumario de la invención

La presente invención proporciona un procedimiento de intra-predicción único que mejora la eficacia de la codificación de vídeo. H.264/AVC usa píxeles de referencia en un límite horizontal ubicado inmediatamente por encima de un bloque objetivo que va a predecirse y píxeles de referencia en un límite vertical ubicado inmediatamente a la izquierda del bloque objetivo. En la presente invención, se recupera al menos parte de o bien una matriz de píxeles de límite horizontal o bien una matriz de píxeles de límite vertical. Después, se añaden los píxeles recuperados a los otros píxeles de límite para extender la matriz de los mismos. Se realiza intra-predicción, basándose únicamente en la matriz extendida de píxeles de límite. En una realización de la presente invención, al menos algunos de los píxeles de límite vertical se recuperan y se añaden a los píxeles de límite horizontal para extender la matriz de los mismos.

La presente invención elimina el procedimiento de decisión de seleccionar o bien el límite horizontal o bien el límite vertical de los que se retiran píxeles de referencia. La presente invención también elimina el procedimiento recurrente de calcular una posición del límite vertical que interseca con una dirección de predicción, en el que el procedimiento de cálculo recurrente normalmente incluye una operación de división. La eliminación de estos procedimientos permite que el procedimiento de intra-predicción se implemente en arquitecturas de una instrucción, múltiples datos (SIMD), mejorando así la eficacia computacional de la codificación de vídeo.

La presente invención proporciona un método de codificación de vídeo que incluye las características según la reivindicación 1 y un descodificador de vídeo que incluye las características según la reivindicación 2. Los ejemplos adicionales denominados realizaciones en la descripción son ejemplos ilustrativos.

Breve descripción de los dibujos

La figura 1 es un diagrama de bloques que muestra una arquitectura de hardware a modo de ejemplo en la que puede implementarse la presente invención.

La figura 2 es un diagrama de bloques que muestra una vista general de un codificador de vídeo al que se le puede aplicar la presente invención.

La figura 3 es un diagrama de bloques que muestra una vista general de un descodificador de vídeo al que se le puede aplicar la presente invención.

La figura 4 es un diagrama de bloques que muestra los módulos funcionales de un codificador según un modo de realización de la presente invención.

La figura 5 es un diagrama de flujo que muestra un procedimiento de intra-predicción realizado por un módulo de intra-predicción del modo de realización de la presente invención.

La figura 6 es un diagrama de bloques que muestra los módulos funcionales de un descodificador según un modo de realización de la presente invención.

La figura 7 es un diagrama que muestra direcciones de predicción que ilustran modos de predicción intra_4x4 soportados en H.264/AVC.

La figura 8 es un diagrama que muestra las direcciones de predicción propuestas en el documento n.º JCT-VC A119.

La figura 9 es un diagrama de flujo que muestra el procedimiento, propuesto en el documento JCT-VC A119, de generación de un bloque predicho a lo largo de una de las direcciones de predicción mostradas en la figura 7.

La figura 10 es un diagrama de flujo que muestra el procedimiento de intra-predicción de baja complejidad realizado según un modo de realización de la presente invención.

La figura 11A es una vista esquemática que muestra un bloque de predicción y matrices de píxeles de límite horizontal y vertical.

La figura 11B es una vista esquemática que muestra una matriz de píxeles de límite horizontal extendida con píxeles de límite vertical.

La figura 12 es un diagrama de flujo que muestra el procedimiento de extender una matriz de píxeles de límite horizontal realizado según un modo de realización de la presente invención.

La figura 13 es un diagrama de flujo que muestra otro modo de realización de extender una matriz de píxeles de límite horizontal.

La figura 14 un diagrama de flujo que muestra el procedimiento de intra-predicción de baja complejidad realizado según otro modo de realización de la presente invención.

Descripción detallada de los dibujos y los modos de realización actualmente preferidos

La figura 1 muestra una arquitectura de hardware a modo de ejemplo de un ordenador 100 en el que puede implementarse la presente invención. Obsérvese que la arquitectura de hardware mostrada en la figura 1 puede ser común tanto en un codificador de vídeo como en un descodificador de vídeo que implementan los modos de realización de la presente invención. El ordenador 100 incluye un procesador 101, memoria 102, dispositivo de almacenamiento 105 y uno o más dispositivos de entrada y/o salida (E/S) 106 (o periféricos) que están acoplados en comunicación a través de una interfaz 107 local. La interfaz 105 local puede ser, por ejemplo, pero sin limitación, uno o más buses u otras conexiones por cable o inalámbricas, tal como se conoce en la técnica.

El procesador 101 es un dispositivo de hardware para ejecutar software, particularmente el almacenado en la memoria 102. El procesador 101 puede ser cualquier procesador fabricado a medida o comercialmente disponible, una unidad de procesamiento central (CPU), un procesador auxiliar entre varios procesadores asociados con el ordenador 100, un microprocesador basado en semiconductor (en forma de un microchip o conjunto de chips) o generalmente cualquier dispositivo para ejecutar instrucciones de software.

La memoria 102 comprende un medio legible por ordenador que puede incluir uno cualquiera o una combinación de elementos de memoria volátil (por ejemplo, memoria de acceso aleatorio (RAM, tal como DRAM, SRAM, SDRAM, etc.)) y elementos de memoria no volátil (por ejemplo, ROM, disco duro, cinta, CD-ROM, etc.). Además, la memoria 102 puede incorporar medios de almacenamiento electrónicos, magnéticos, ópticos y/o de otros tipos. Un medio legible por ordenador puede ser cualquier medio que pueda almacenar, comunicar, propagar o transportar el programa para su uso por o en conexión con el sistema, aparato o dispositivo de ejecución de instrucciones. Obsérvese que la memoria 102 puede tener una arquitectura distribuida, en la que diversos componentes están situados alejados unos de otros, pero a los que puede acceder el procesador 101.

El software 103 en la memoria 102 puede incluir uno o más programas separados, cada uno de los cuales contiene una lista ordenada de instrucciones ejecutables para implementar funciones lógicas del ordenador 100, tal como se

describe a continuación. En el ejemplo de la figura 1, el software 103 en la memoria 102 define la funcionalidad de codificación de vídeo o descodificación de vídeo del ordenador 100 según la presente invención. Además, aunque no se requiere, es posible que la memoria 102 contenga un sistema operativo (S/O) 104. El sistema operativo 104 controla esencialmente la ejecución de programas informáticos y proporciona planificación, control de entrada-salida, gestión de archivos y datos, gestión de memoria y control de comunicación y servicios relacionados.

El dispositivo de almacenamiento 105 del ordenador 100 puede ser uno de muchos tipos diferentes de dispositivo de almacenamiento, incluyendo un dispositivo de almacenamiento estacionario o dispositivo de almacenamiento portátil. Como ejemplo, el dispositivo de almacenamiento 105 puede ser una cinta magnética, disco, memoria flash, memoria volátil o un dispositivo de almacenamiento diferente. Además, el dispositivo de almacenamiento 105 puede ser una tarjeta de memoria digital segura o cualquier otro dispositivo de almacenamiento 105 extraíble.

Los dispositivos de E/S 106 pueden incluir dispositivos de entrada, por ejemplo, pero sin limitación, una pantalla táctil, un teclado, ratón, escáner, micrófono u otro dispositivo de entrada. Además, los dispositivos de E/S 106 también pueden incluir dispositivos de salida, por ejemplo, pero sin limitación, una pantalla u otros dispositivos de salida. Los dispositivos de E/S 106 pueden incluir además dispositivos que se comunican a través tanto de entradas como de salidas, por ejemplo, pero sin limitación, un modulador/desmodulador (módem; para acceder a otro dispositivo, sistema o red), un transceptor de radiofrecuencia (RF), inalámbrico u otro, una interfaz telefónica, un puente, un enrutador u otros dispositivos que funcionan como entrada y como salida.

Tal como conocen bien los expertos habituales en la técnica, la compresión de vídeo se logra eliminando información redundante en una secuencia de vídeo. Existen muchas normas diferentes de codificación de vídeo, ejemplos de las cuales incluyen MPEG-1, MPEG-2, MPEG-4, H.261, H.263 y H.264/AVC. Debe observarse que no se pretende limitar la presente invención en cuanto a la aplicación de cualquier norma de codificación de vídeo específica. Sin embargo, la siguiente descripción de la presente invención se proporciona usando el ejemplo de la norma H.264/AVC. H.264/AVC es la norma de codificación de vídeo más reciente y logra una mejora de rendimiento significativa con respecto a las normas de codificación anteriores tales como MPEG-1, MPEG-2, H.261 y H.263.

En H.264/AVC, cada trama o imagen de un vídeo puede descomponerse en varios segmentos. Los segmentos se dividen entonces en bloques de 16x16 píxeles denominados macrobloques, que después pueden dividirse adicionalmente en bloques de 8x16, 16x8, 8x8, 4x8, 8x4, hasta 4x4 píxeles. Hay cinco tipos de segmentos soportados por H.264/AVC. En los segmentos I, todos los macrobloques se codifican usando intra-predicción. En los segmentos P, los macrobloques pueden codificarse usando intra o inter-predicción. Los segmentos P solo permiten usar una señal de predicción compensada por movimiento (MCP) por macrobloque. En los segmentos B, pueden codificarse macrobloques usando intra o inter-predicción. Pueden usarse dos señales de MCP por predicción. Los segmentos SP permiten conmutar segmentos P entre diferentes flujos de vídeo de manera eficaz. Un segmento SI es una coincidencia exacta para un segmento SP para acceso aleatorio o recuperación de error, mientras que solo se usa intra-predicción.

La figura 2 muestra una vista general de un codificador de vídeo al que se le puede aplicar la presente invención. Los bloques mostrados en la figura representan módulos funcionales realizados por el procesador 101 que ejecuta el software 103 en la memoria 102. Se alimenta una imagen 200 de trama de vídeo a un codificador de vídeo 201. El codificador de vídeo trata la imagen 200 en unidades de macrobloques 200A. Cada macrobloque contiene varios píxeles de imagen 200. En cada macrobloque se realiza una transformación en coeficientes de transformación seguida por una cuantificación en niveles de coeficientes de transformación. Además, se usa intra-predicción o inter-predicción, para no realizar las etapas de codificación directamente en los datos de píxel sino en las diferencias de los mismos con respecto a valores de píxel predichos, logrando así valores pequeños que se comprimen más fácilmente.

Para cada segmento, el codificador 201 genera varios elementos de sintaxis, que forman una versión codificada de los macrobloques del segmento respectivo. Todos los elementos de datos residuales en los elementos de sintaxis, que están relacionados con la codificación de coeficientes de transformación, tales como los niveles de coeficientes de transformación o un mapa de significación que indica niveles de coeficientes de transformación omitidos, se denominan elementos de sintaxis de datos residuales. Además de estos elementos de sintaxis de datos residuales, los elementos de sintaxis generados por el codificador 201 contienen elementos de sintaxis de información de control que contienen información de control sobre cómo se ha codificado cada macrobloque y cómo tiene que descodificarse, respectivamente. En otras palabras, los elementos de sintaxis pueden dividirse en dos categorías. La primera categoría, los elementos de sintaxis de información de control, contiene los elementos relacionados con un tipo de macrobloque, tipo de sub-macrobloque e información sobre modos de predicción de tipos tanto espacial como temporal, así como información de control basada en segmento y basada en macrobloque, por ejemplo. En la segunda categoría, todos los elementos de datos residuales, tales como un mapa de significación que indica las ubicaciones de todos los coeficientes significativos dentro de un bloque de coeficientes de transformación cuantificados y los valores de los coeficientes significativos, que se indican en unidades de niveles correspondientes a las etapas de cuantificación, se combinan y se convierten en elementos de sintaxis de datos residuales.

El codificador 201 comprende un codificador de entropía que codifica elementos de sintaxis y genera contraseñas

aritméticas para cada segmento. Cuando se generan las contraseñas aritméticas para un segmento, el codificador de entropía aprovecha dependencias estadísticas entre los valores de datos de elementos de sintaxis en el flujo de bits de la señal de vídeo. El codificador 201 emite una señal de vídeo codificada para un segmento de imagen 200 a un descodificador de vídeo 301 mostrado en la figura 3.

La figura 3 muestra una vista general de un descodificador de vídeo al que se le puede aplicar la presente invención. Asimismo, los bloques mostrados en la figura representan módulos funcionales realizados por el procesador 101 que ejecuta el software 103 en la memoria 102. El descodificador de vídeo 301 recibe la señal de vídeo codificada y en primer lugar realiza la descodificación de entropía de la señal de vuelta a los elementos de sintaxis. El descodificador 301 usa los elementos de sintaxis para reconstruir, macrobloque por macrobloque y después segmento por segmento, las muestras 300A de imagen de píxeles en la imagen 300.

La figura 4 muestra los módulos funcionales del codificador de vídeo 201. Estos módulos funcionales se realizan mediante el procesador 101 que ejecuta el software 103 en la memoria 102. Una imagen de vídeo de entrada es una trama o un campo de una imagen de vídeo natural (sin comprimir) definida por puntos de muestra que representan componentes de colores originales, tales como crominancia ("croma") y luminancia ("luma") (otras componentes son posibles, por ejemplo, tono, saturación y valor). La imagen de vídeo de entrada se divide en macrobloques 400 que representan cada uno un área de imagen cuadrada que consiste en 16x16 píxeles de la componente luma del color de la imagen. La imagen de vídeo de entrada también se reparte en macrobloques que representan cada uno 8x8 píxeles de cada una de las dos componentes de croma del color de la imagen. En el funcionamiento de codificador general, los macrobloques introducidos pueden predecirse de manera temporal o espacial usando inter o intra-predicción. Sin embargo, con el propósito de discusión, se supone que los macrobloques 400 son todos macrobloques de tipo segmento I y se someten únicamente a intra-predicción.

La intra-predicción se logra en un módulo de intra-predicción 401, cuyo funcionamiento se analizará con detalle a continuación. El módulo de intra-predicción 401 genera un bloque de predicción 402 a partir de píxeles de límite horizontal y vertical de bloques adyacentes, que se han codificado, reconstruido y almacenado anteriormente en una memoria de trama 403. Un residuo 404 del bloque de predicción 402, que es la diferencia entre un bloque objetivo 400 y el bloque de predicción 402, se transforma, ajusta a escala y cuantifica en un módulo de transformación/cuantificación 405, usando métodos y técnicas conocidas por los expertos en el campo de la codificación de vídeo. Entonces se someten los coeficientes de transformación cuantificados 406 a codificación de entropía en un módulo de codificación de entropía 407 y se transmiten (junto con otra información relacionada con la intra-predicción) como una señal de vídeo codificada 408.

El codificador de vídeo 201 contiene funcionalidad de descodificación para realizar la intra-predicción en bloques objetivo. La funcionalidad de descodificación comprende un módulo de cuantificación/transformación inverso 409, que realiza la cuantificación inversa y la transformación inversa en los coeficientes de transformación cuantificados 406 para producir el residuo de predicción descodificado 410, que se añade al bloque de predicción 402. La suma del residuo de predicción descodificado 410 y el bloque de predicción 402 es un bloque reconstruido 411, que se almacena en la memoria de trama 403 y se leerá de la misma y será utilizado por el módulo de intra-predicción 401 para generar un bloque de predicción 402 para descodificar un siguiente bloque objetivo 400.

La figura 5 es un diagrama de flujo que muestra procedimientos realizados por el módulo de intra-predicción 401. Según la norma H.264/AVC, la intra-predicción implica predecir cada píxel del bloque objetivo 400 en una pluralidad de modos de predicción, usando interpolaciones de píxeles de límite ("píxeles de referencia") de bloques adyacentes anteriormente codificados y reconstruidos. Los modos de predicción se identifican mediante números enteros positivos 0, 1, 2..., cada uno asociado con una instrucción o un algoritmo diferente para predecir píxeles específicos en el bloque objetivo 400. El módulo de intra-predicción 401 ejecuta una intra-predicción en los modos de predicción respectivos y genera diferentes bloques de predicción. En un algoritmo de búsqueda completa ("FS"), cada uno de los bloques de predicción generados se compara con el bloque objetivo 400 para encontrar el modo de predicción óptimo, lo cual minimiza el residuo de predicción 404 o produce un residuo de predicción 404 menor entre los modos de predicción. La identificación del modo de predicción óptimo se comprime y se envía al descodificador 301 con otros elementos de sintaxis de información de control.

Cada modo de predicción puede describirse por una dirección general de predicción tal como se describe verbalmente (es decir, horizontal hacia arriba, vertical y diagonal hacia abajo y a la izquierda). Una dirección de predicción puede describirse gráficamente mediante una dirección angular que se expresa a través de un diagrama con flechas tal como se muestra en la figura 7. En este tipo de diagrama, puede considerarse que cada flecha representa una dirección de predicción o un modo de predicción. El ángulo correspondiente a un modo de predicción tiene una relación general con respecto a la dirección desde la ubicación promedio ponderada de los píxeles de referencia usados para predecir un píxel objetivo en la ubicación de píxel objetivo. Obsérvese que los modos de predicción incluyen un modo de predicción DC que no está asociado con ninguna dirección de predicción y, por tanto, no puede describirse gráficamente en el diagrama al contrario que los demás modos de predicción. En el modo de predicción DC, el bloque de predicción 402 se genera de tal manera que cada píxel en el bloque de predicción 402 se establece uniformemente al valor medio de los píxeles de referencia.

Volviendo a la figura 5, el modo de predicción se inicia en la etapa 501. Entonces se determina, en la etapa 502, si el modo de predicción indica la predicción DC. Si es así, el flujo avanza a la etapa 503, en la que se genera un bloque de predicción 402 DC con el valor medio de los píxeles de referencia en la etapa 503. Si el modo de predicción indica otra cosa, se genera un bloque de predicción 402 según la instrucción o el algoritmo asociado con el modo de predicción en la etapa 504, cuyo procedimiento se analizará en detalle a continuación. Tras la etapa 503 o 504, el flujo avanza a la etapa 505, en la que se determina si los bloques de predicción se generan para todos los modos de predicción. Si se ejecuta intra-predicción en todos los modos de predicción, el flujo avanza a la etapa 506. De lo contrario, el modo de predicción se aumenta en la etapa 507 y el flujo vuelve a la etapa 502. En la etapa 506, se compara cada uno de los bloques de predicción generados con el bloque objetivo 400 para determinar el modo de predicción óptimo, que minimiza el residuo de predicción 404.

La figura 6 muestra los módulos funcionales del descodificador de vídeo 301. Estos módulos funcionales se realizan mediante el procesador 101 que ejecuta el software 103 en la memoria 102. La señal de vídeo codificada del codificador 201 es recibida en primer lugar por un descodificador de entropía 600 y se somete a descodificación de entropía para obtener de nuevo coeficientes de transformación cuantificados 601. Los coeficientes de transformación cuantificados 601 se someten a cuantificación inversa y se transforman mediante un módulo de cuantificación/transformación inverso 602 para generar un residuo de predicción 603. Se notifica a un módulo de intra-predicción 604 del modo de predicción seleccionado por el codificador 201. Según el modo de predicción seleccionado, el módulo de intra-predicción 604 realiza un procedimiento de intra-predicción similar al realizado en las etapas 502, 503 y 504 de la figura 5 para generar un bloque de predicción 605, usando píxeles de límite de bloques adyacentes anteriormente reconstruidos y almacenados en una memoria 606 de trama. El bloque de predicción 605 se añade al residuo de predicción 603 para reconstruir un bloque de señal de vídeo descodificada 607. El bloque 607 reconstruido se almacena en la memoria 606 de trama para su uso en la predicción de un bloque siguiente.

Se facilitará una descripción detallada de la siguiente manera sobre el procedimiento de la etapa 504 realizado por los módulos de intra-predicción 401 y 604 para generar un bloque de predicción en uno de los modos de predicción, excepto el modo de predicción DC. H.264/AVC soporta predicción intra_4x4, predicción intra_8x8 y predicción intra_16x16. La predicción intra_4x4 se usa comúnmente cuando hay un detalle significativo en la imagen. La predicción intra_4x4 predice los dieciséis bloques de luma 4x4 dentro de un macrobloque de manera individual. La predicción intra_4x4 se realiza en nueve modos de predicción, incluyendo un modo de predicción DC. Las direcciones de predicción espacial a lo largo de las cuales se realiza la predicción intra_4x4 se muestran en la figura 7. La predicción intra_8x8 se realiza en nueve modos de predicción, incluyendo un modo de predicción DC. La predicción intra_16x16 se realiza en cuatro modos de predicción, incluyendo un modo de predicción DC.

Estudios recientes muestran que un aumento en el número de direcciones de predicción o un aumento en el número de modos de predicción, contribuye generalmente a mejorar la eficacia de compresión en la codificación de vídeo. Véanse, por ejemplo, los documentos n.os JCT-VC A119 (“Angular intra prediction”) y JCT-VC A124 (“Arbitrary direction intra”) presentados al Joint Collaborative Team on Video Coding (JCT-VC). Un aumento en el número de direcciones de predicción conduce a un aumento en el número de intervalos angulares de direcciones de predicción disponibles y, por tanto, a un aumento en el número de candidatos de bloque de predicción. El número aumentado de candidatos de bloque de predicción simplemente aumenta las posibilidades de tener un bloque de predicción que sea casi el mismo que un bloque objetivo que va a codificarse. La figura 8 es un diagrama que muestra las direcciones de predicción propuestas en el documento n.º JCT-VC A119. En la figura 8, los píxeles de referencia consisten en diecisiete (17) píxeles horizontales y diecisiete (17) píxeles verticales, en los que el píxel superior izquierdo es común a los límites tanto horizontal como vertical. Por tanto, hay 33 direcciones de predicción diferentes disponibles para generar píxeles de predicción en un bloque 8x8. JCT-VC A124 propone una intra-predicción direccional arbitraria en la que el número de direcciones de predicción se ajusta según el tamaño de un bloque que va a predecirse.

La figura 9 es un diagrama de flujo que muestra el procedimiento, propuesto en el documento JCT-VC A119, de generar un bloque de predicción a lo largo de una de las direcciones de predicción mostradas en la figura 8. En la siguiente descripción del procedimiento, algunos algoritmos se simplifican para facilitar la explicación. Además, el procedimiento descrito se limita a la intra-predicción a lo largo de una dirección de predicción que es principalmente vertical. La intra-predicción a lo largo de una dirección de predicción que es principalmente horizontal, puede implementarse de manera simétrica al procedimiento mostrado en la figura 9, tal como se demuestra en el software proporcionado por el documento JCT-VC A119. Aunque la figura 8 muestra un bloque 8x8 que va a predecirse, el procedimiento mostrado en la figura 9 puede expandirse para aplicarse a diversos números de píxeles en diferentes configuraciones. Por ejemplo, un bloque que va a predecirse puede comprender una matriz 4x4 de píxeles. Un bloque de predicción también puede comprender una matriz 8x8 de píxeles, una matriz 16x16 de píxeles o matrices más grandes de píxeles. Otras configuraciones de píxeles, incluyendo matrices tanto cuadradas como rectangulares, también pueden constituir un bloque de predicción.

En la etapa 900 en la figura 9, se leen píxeles de referencia en límites horizontal y vertical, que se encuentran inmediatamente por encima y a la izquierda de un bloque objetivo, respectivamente, a partir de bloques adyacentes que se han codificado, reconstruido y almacenado anteriormente en una memoria de trama, tal como la memoria

403 mostrada en la figura 4. Los píxeles del límite horizontal se almacenan en un área de memoria denominada *"refH".* Los píxeles del límite vertical se almacenan en otra área de memoria denominada *"refV".* Volviendo a la figura 8, los píxeles de referencia se identifican mediante sus coordenadas en un sistema de coordenadas que tiene el origen en la posición de píxel superior izquierda en el bloque 8x8. Por tanto, los píxeles de límite horizontal tienen coordenadas expresadas por p[x, y] con x = 0, 1...16 e y = 0. Los píxeles de límite vertical tienen coordenadas expresadas por p[x, y] con x = 0, y = 0, -1, -2...-16.

Se supone que los píxeles de límite horizontal almacenados en el área de memoria *refH* se identifican mediante una dirección lógica (x) con x = 0, 1...16 y que los píxeles de límite vertical almacenados en el área de memoria *refV* se identifican igualmente mediante una dirección lógica (y) con y = 0, -1, -2...-16, donde cada píxel se almacena en la dirección que tiene el número en la coordenada de la cual se lee. Por tanto, a medida que se representan gráficamente los píxeles horizontales y verticales en la figura 8, puede considerarse que las áreas de memoria *refH* y *refV* se extienden de manera lineal y ortogonal entre sí y que tienen, cada una, una longitud de 2 x *size* + 1, donde *"size"* es un parámetro que representa el tamaño del bloque objetivo. Se supone que "*size"* tiene un valor igual a una potencia entera de 2, tal como 4, 8, 16... Opcionalmente puede aplicarse un filtro de paso bajo, tal como se describe en la sección 8.3.2.2.1 en H.264/AVC, a los píxeles en *refH* y *refV*.

En la etapa 901, se establece un contador denominado "*row*" a cero ("0"). El contador *row* adopta un valor de desde 0 hasta *size* e indica una posición de fila de un píxel de predicción en el bloque de predicción. En la etapa 902, se calcula un parámetro denominado "*pos*" mediante *angle* X (*row*+1). *angle* es un parámetro que tiene un número fraccionario en una representación de puntos fijos. Como tal, *angle* está formado por una parte entera y una parte fraccionaria, y la parte fraccionaria consiste en un número fijado de dígitos binarios. *angle* representa una de las direcciones de predicción mostradas en la figura 8. Por ejemplo, "*angle* = *-size*" identifica la dirección de predicción que pasa a través de las coordenadas [x = 0, y = 0] en la figura 8. Un *angle* que tiene un valor positivo identifica una dirección de predicción que interseca únicamente el límite horizontal, mientras que un *angle* que tiene un valor negativo identifica una dirección de predicción que interseca los límites tanto horizontal como vertical. *angle* varía dentro de un intervalo determinado por el número de direcciones de predicción deseadas que van a usarse. Tal como se propone en el documento JCT-VC A124, el número de direcciones de predicción que van a usarse puede determinarse según el tamaño de un bloque que va a predecirse. En la siguiente descripción, se supone que *angle* adopta un número fraccionario que varía dentro de un intervalo desde "-*size*" hasta "*size*". Obsérvese que los límites de intervalo de *angle* pueden definirse con otros valores.

Al igual que *angle,* el parámetro *pos* consiste en una parte entera y una parte fraccionaria, y la parte fraccionaria del mismo consiste en un número fijado de dígitos binarios, que es igual al logaritmo en base 2 del límite de intervalo de *angle,* que puede expresarse mediante *log2_size* según la suposición anterior de que el límite de intervalo de *angle* se establece al *size. pos* identifica la posición de una intersección entre el límite horizontal y la dirección de predicción representada por *angle.* Volviendo a la etapa 902, la operación "*pos* >> *log2_size*" identifica un número entero número en *pos,* que se almacena en un parámetro "*int*", y la operación "*pos* & (*size* - 1)" identifica un número fraccionario en *pos,* que se almacena en un parámetro "*frac*". El operador ">>" representa un desplazamiento aritmético a la derecha de dígitos binarios. El operador "&" representa la operación "y" relacionada con los bits.

En la etapa 903, se determina si *angle* tiene un valor igual o superior a cero ("0"). Si *angle* tiene un valor igual o superior a cero, el flujo avanza a la etapa 904. De lo contrario, el flujo avanza a la etapa 913. *angle* igual o superior a cero sugiere que solo es posible basarse en los píxeles de referencia ubicados en el límite horizontal o almacenados en *refH,* para obtener píxeles de predicción en un bloque de predicción. Por otro lado, *angle* inferior a cero sugiere que se necesitan píxeles de referencia ubicados en el límite vertical o almacenados en *refV,* para obtener píxeles de predicción en el bloque de predicción.

En la etapa 904, se determina si *frac* es distinto de cero. Si *frac* es distinto de cero, el flujo avanza a la etapa 905. Si *frac* es cero, el flujo avanza a la etapa 906. *frac* igual a cero sugiere que puede copiarse un píxel de predicción en el bloque de predicción directamente de un píxel de referencia en el límite horizontal. *frac* distinto de cero sugiere que la dirección de predicción interseca el límite horizontal en una posición distinta de un número entero, y se necesita una interpolación de más de un píxel de referencia para obtener un píxel de predicción en el bloque de predicción.

En la etapa 905, un contador denominado "*col*" se establece a cero ("0"). El contador *col* se usa para abordar un píxel de referencia en *refH.* En la etapa 907, se recuperan dos píxeles de referencia identificados por *"int* + *col* + 1" e *"int* + *col* + 2" de *refH.* Se calcula el promedio ponderado de estos dos píxeles de referencia o se interpolan con *frac* para obtener un píxel de predicción *v*. Específicamente, se multiplica un píxel de referencia en *refH* identificado por *"int* + *col* + 1" por "*size* - *frac*" y se almacena en un parámetro *a.* Se multiplica un píxel de referencia en *refH* identificado por *"int* + *col* + 2" por "*frac*" y se almacena en un parámetro *b.* Después se suman los parámetros *a* y *b* y se dividen entre *size,* es decir, (*size* - *frac*) + *frac.* La división entre *size* puede sustituirse por desplazamiento a la derecha mediante *log2_size*. El píxel de predicción obtenido *v* se almacena en una matriz de áreas de memoria denominada "*pred*", que representa un bloque de predicción para el bloque objetivo en una dirección de predicción particular. Cada área de memoria en *pred* se identifica mediante los parámetros *row* y *col.* Después, se aumenta *col* en 1 en la etapa 908 y se compara con *size* en la etapa 909. Siempre que *col* sea menor que *size,* se repiten las etapas 907 y 908. Cuando *col* se vuelve igual a *size,* el flujo avanza a la etapa 920.

Si se determina que *frac* es cero en la etapa 904, el contador *col* se establece a cero en la etapa 906. En la etapa 910, se copia el píxel de predicción *v* directamente de *refH* (*int* + *col* + 1) y después se almacena en el área de memoria correspondiente en *pred.* Entonces se aumenta *col* en 1 en la etapa 911 y se compara con *size* en la etapa 912. Siempre que *col* sea menor que *size,* se repiten las etapas 910 y 911. Cuando *col* se vuelve igual a *size,* el flujo avanza a la etapa 920.

Volviendo a la etapa 903, *angle* inferior a cero requiere píxeles de referencia de *refV* para obtener píxeles de predicción en el bloque de predicción. El contador *col* se establece a cero en la etapa 913. Entonces se determina, en la etapa 914, si *"int* + *col* + 1" es inferior a cero. *"int* + *col* + 1" igual o superior a cero sugiere que todavía solo es posible basarse en los píxeles de referencia almacenados en *refH* para obtener píxeles de predicción en el bloque de predicción y el flujo avanza a la etapa 915. El procedimiento realizado en la etapa 915 es similar al de la etapa 907, y no se repetirá la descripción del mismo aquí. Entonces se aumenta *col* en 1 en la etapa 916 y se compara con *size* en la etapa 917. Siempre que *col* sea menor que *size,* se repiten las etapas 914, 915 y 916. Cuando *col* se vuelve igual a *size,* el flujo avanza a la etapa 920.

Si se determina que *"int* + *col* + 1" es inferior a cero en la etapa 914, se necesitan píxeles de referencia almacenados en *refV* para obtener píxeles de predicción en el bloque de predicción. En la etapa 918, en primer lugar se determina la posición de una intersección entre el límite vertical y una dirección de predicción. En la etapa 918, la posición se representa mediante *pos2.* Obsérvese que en la etapa 902, *pos,* es decir, la posición de una intersección entre el límite horizontal y una dirección de predicción, se determina mediante "*angle* X (*row* + 1)". Dado que *angle* representa una proporción de diferencias horizontal y vertical, se calcula "$angle^{-1}$ X (*col* + 1)", en lugar de "*angle* x (*row* + 1)", para determinar la posición de una intersección entre el límite vertical y una dirección de predicción. Tal como se supuso anteriormente, *angle* está dentro del intervalo de *-size* a *size* (*-size* ≤ *angle* ≤ *size*). Por tanto, una proporción α entre *angle* y *size* se define mediante:

$$\alpha = \frac{angle}{size} \ (-1 \leq \alpha \leq 1).$$

Entonces, $angle^{-1}$ se define mediante:

$$angle^{-1} = \frac{size}{\alpha} \text{ o } \frac{size^2}{angle}.$$

Como tal, *pos2* se determina en la etapa 918 con el cuadrado de *size* multiplicado por *col* + 1 y después dividido entre el valor absoluto de *angle* de la siguiente manera:

$$pos2 = \frac{size^2 \times (col+1)}{|angle|}.$$

Al igual que *pos, pos2* tiene un número fraccionario en una representación de puntos fijos que está formado por una parte entera y una parte fraccionaria. La parte fraccionaria consiste en el número de dígitos binarios determinados por *log2_size.* La parte entera de *pos2* se almacena en un parámetro *int2* y la parte fraccionaria de *pos2* se almacena en un parámetro *frac2.* En la etapa 919, se recuperan dos píxeles de referencia identificados mediante *"int2* + *row* + 1" e "*int2* + *row* + 2" de *refV.* Se calcula el promedio ponderado de estos dos píxeles de referencia o se interpolan con *frac2* para obtener un píxel de predicción v. Específicamente, se multiplica un píxel de referencia de *refV* (*int2* + *row* + 1) por "*size - frac2*" y se almacena en un parámetro *a.* Se multiplica un píxel de referencia de *refV* (*int2* + *row* + 2) por "*frac2*" y se almacena en un parámetro *b.* Entonces se suman los parámetros *a* y *b* y se dividen entre *size* o se desplazan a la derecha mediante *log2_size*. El píxel de predicción obtenido v se almacena en el área de memoria correspondiente de *pred*. Se repiten las etapas 914, 918, 919 y 916 hasta que *col* se vuelve igual a *size* en la etapa 917.

En la etapa 920, se aumenta *row* en 1. Entonces se determina, en la etapa 921, si *row* es menor que *size.* Siempre que *row* sea menor que *size,* se repiten las etapas desde la etapa 902 para obtener un píxel de predicción en el bloque de predicción. El flujo termina cuando *row* se vuelve igual a *size* en la etapa 921.

Tal como se ha mencionado anteriormente, un aumento en el número de candidatos de bloque de predicción contribuye a mejorar la eficacia de codificación, mientras que un aumento en el número de candidatos de bloque de predicción conduce a un aumento en la carga de trabajo computacional. Por lo tanto, con el fin de aumentar el número de candidatos de bloque de predicción para así mejorar la eficacia de codificación, se necesita revisar el procedimiento de generación de un candidato de bloque de predicción para lograr mayor eficacia del procedimiento. Al revisar el procedimiento mostrado en la figura 9, pueden identificarse dos cuellos de botella computacionales. El

primer cuello de botella computacional es la operación de comparación y ramificación de la etapa 914, que se repite dentro del bucle. El segundo cuello de botella computacional es la operación de división de la etapa 918, que también se repite dentro del bucle.

En la actualidad, se dispone de arquitecturas de una instrucción, múltiples datos (SIMD) para un cálculo eficaz. SIMD permite que ordenadores con múltiples elementos de procesamiento realicen la misma operación con múltiples datos simultáneamente. Sin embargo, las arquitecturas SIMD típicas no soportan la implementación de división y cálculo/ramificación en un bucle y, por lo tanto, no pueden usarse para implementar el procedimiento mostrado en la figura 9 debido a la inclusión de las etapas 914 y 918 en el bucle, aunque los bucles que comienzan en las etapas 907 y 910 son lo suficientemente robustos como para implementarse con SIMD. Por lo tanto, un objetivo de la presente invención es eliminar los cuellos de botella computacionales del procedimiento mostrado en la figura 9 y proporcionar intra-predicción de baja complejidad, que permite que arquitecturas SIMD típicas implementen procesamiento en paralelo a lo largo de todas las direcciones de predicción mostradas en la figura 8.

La figura 10 es un diagrama de flujo que muestra el procedimiento de intra-predicción de baja complejidad según un modo de realización de la presente invención, que está diseñado para sustituir al procedimiento de la figura 9 en la implementación del procedimiento en la etapa 504 de la figura 5. En la figura 10, las mismas etapas de procedimiento que las realizadas en la figura 9 se identifican con los mismos números de etapa que los usados en la figura 9, tales como las etapas 900, 901, 902, 904, 905, 906, 907, 908, 909, 910, 911, 912, 920 y 921. La descripción de estas etapas comunes no se repite aquí. Las etapas 1000 y 1001 son etapas particulares del procedimiento de la figura 10. Tal como resulta evidente a partir de una comparación con el procedimiento mostrado en la figura 9, el procedimiento de la figura 10 elimina la etapa de comparación de la etapa 903 y todas las etapas que se ramifican hacia la izquierda desde la etapa 903, que se realizan cuando *angle* es inferior a cero, eliminando así los cuellos de botella computacionales de las etapas 914 y 918.

En las etapas 1000 y 1001 añadidas, se determina si *angle* es igual o superior a -1. Cuando *angle* es igual o superior a -1, los píxeles de referencia ubicados en el límite horizontal son suficientes para generar un píxel de predicción en el bloque de predicción, y no se necesitan los píxeles de referencia en el límite vertical. Por otro lado, cuando *angle* es inferior a -1, se necesitan píxeles de referencia en el límite vertical para generar un píxel de predicción en el bloque de predicción. En la etapa 1001, se extienden píxeles de referencia almacenados en *refH* en la dirección negativa, usando al menos algunos de los píxeles almacenados en *refV*. Las figuras 11A y 11B son representaciones esquemáticas que muestran la extensión de *refH* realizada en la etapa 1001. En la figura 11A, los píxeles de referencia 1102 almacenados en *refH* son del límite horizontal ubicado por encima del bloque objetivo 1101. Los píxeles de referencia 1103 almacenados en *refV* son del límite vertical ubicado a la izquierda del bloque objetivo 1101. Tal como se muestra en la figura 11B, tras la etapa 1001 de la figura 10, algunos de los píxeles de referencia en *refV* se copian en *refH*, y *refH* tiene una parte extendida 1104 que se extiende en la dirección negativa.

La figura 12 es un diagrama de flujo que muestra detalles del procedimiento realizado en la etapa 1001. En la etapa 1201, se establece un contador *col* a -1. Se usa *col* para identificar una dirección de la parte extendida de *refH*. En la etapa 1202, un píxel de referencia en *refV* que va a copiarse en la parte extendida de *refH* se identifica mediante:

$$\frac{size \times col}{angle}.$$

La división en la ecuación anterior es una división de números enteros y la ecuación produce un número entero. La ecuación funciona de manera similar al procedimiento de la etapa 918 mostrada en la figura 9. En la etapa 918, se calcula un valor de número entero de *pos2* mediante:

$$\frac{(size^2 \times (col+1))}{angle} >> \log 2\_size.$$

Obsérvese que el desplazamiento a la derecha mediante *log2_size* es equivalente a la división entre *size.*

En la etapa 1203, se reduce *col* en 1. Después se determina, en la etapa 1204, si *col* es igual a *angle*. Si *col* no es igual a *angle,* el flujo vuelve a la etapa 1202. Se repiten las etapas 1202 y 1203 hasta que *col* se vuelve igual a *angle.* Por lo tanto, se leen píxeles de referencia de *refV* en el orden ascendente, o desde la parte superior hasta la parte inferior del límite vertical, y se copian en *refH* también en el orden descendente, o desde la derecha hasta la izquierda del límite horizontal. Además, no todos los píxeles de referencia en *refV* se copian en *refH.* Solo los píxeles de referencia ubicados dentro del intervalo desde la parte superior hasta la intersección de una dirección de predicción se copian de *refV* en *refH*.

Volviendo a la figura 10, las etapas de procedimiento comenzando desde la etapa 902 se copian de la figura 9, e

incluyen las etapas para generar píxeles de predicción ramificadas hacia la derecha desde la etapa de comparación de la etapa 903 en la figura 9. Sin embargo, obsérvese que las etapas en la figura 10 para generar píxeles de predicción usan *refH* extendido (una suma de las partes 1102 + 1104 en la figura 11B), mientras que las etapas correspondientes en la figura 9 usan *refH* original (parte 1102 en la figura 10A). Dado que *refH* se extiende en la dirección negativa, no se necesita una operación de intra-predicción separada diseñada específicamente para usar píxeles de referencia almacenados en *refV*, tal como se ramifica hacia la izquierda desde la etapa 903 en la figura 9, independientemente del signo de *angle.*

La figura 13 es un diagrama de flujo que muestra otro modo de realización del procedimiento para extender *refH,* usando píxeles de referencia en *refV*. El procedimiento mostrado en las figuras 11 y 12 elimina las etapas de cuello de botella de las etapas 914 y 918 mostradas en la figura 9 y, por lo tanto, se espera que mejore la eficacia del procedimiento de intra-predicción. El procedimiento mostrado en la figura 13 elimina la operación de división realizada en la etapa 1202 de la figura 12 del bucle para copiar píxeles de referencia de *refV* en *refH*. Al eliminar la operación de división del bucle, se espera que el procedimiento mostrado en la figura 13 mejore adicionalmente la eficacia del procedimiento de intra-predicción.

El procedimiento mostrado en la figura 13 sustituye la etapa 1202 de la figura 12 por las etapas 1301 y 1302. La etapa 1302 está dentro del bucle para copiar píxeles de referencia de *refV* en *refH,* mientras que la etapa 1301 está fuera del bucle. La etapa 1301 introduce un nuevo parámetro denominado "*InvAngle*". *InvAngle* se define mediante:

$$256 \times \frac{size}{angle}.$$

La multiplicación por 256 es equivalente a un desplazamiento a la izquierda mediante 8 y garantiza que cada bit resultante de la operación de "*size*/*angle*" representa el cálculo de identificar un píxel de referencia en *refV*. En la etapa 1302, la dirección de un píxel de referencia en *refV* que va a copiarse en la parte extendida de *refH* se identifica mediante:

$$col \times InvAngle >> 8 \ .$$

El resultado de "*col* x *InvAngle*" se somete a desplazamiento a la derecha de 8 para deshacer la operación de desplazamiento a la izquierda realizada en la etapa 1301. Obsérvese que la operación de desplazamiento a la derecha en la etapa 1302 funciona para redondear a la baja el resultado de *"col* x *InvAngle*". Para redondear al número entero más próximo, puede añadirse una compensación de redondeo de 128 al resultado de *"col* x *InvAngle*" antes de realizar la operación de desplazamiento a la derecha. Debe observarse que el número "256" solo es un ejemplo, y la etapa 1301 puede adoptar otro número de compensación, preferiblemente una potencia entera de 2, siempre que el número sea lo suficientemente grande como para conservar todos los bits resultantes de la operación de "*size*/*angle*". Por ejemplo, el número puede ser 64 en la etapa 1301, en lugar de 256, y el número de desplazamientos a la derecha es 6 en la etapa 1302, en lugar de 8. Si se adopta 64, la compensación de redondeo debe ser de 32.

El cálculo realizado en la etapa 1301 puede sustituirse por una operación de consulta para reducir adicionalmente la carga de trabajo computacional. En otras palabras, se prepara una tabla de consulta que almacena valores de *InvAngle* en relación con los valores de *angle.* La tabla 1 proporcionada a continuación es una tabla a modo de ejemplo para la consulta en la etapa 1301:

Tabla 1

angle	1	2	3	4	5	6	7	8
InvAngle	2048	1024	683	512	410	341	293	256

Se supone que, en la tabla anterior, *size* es 8, y *angle* adopta valores de número entero de desde 1 hasta 8. Sin embargo, debe observarse que *size* no se limita a 8 y puede adoptar otro valor, tal como 4 y 16. Además, *angle* puede ser un número fraccionario en una representación de puntos fijos tal como se ha definido anteriormente.

Cuando se copia un píxel de referencia de *refV* a *refH* en la etapa 1202 de la figura 12 o la etapa 1302 de la figura 13, el píxel de referencia puede pasar a través de un filtro de paso bajo para reducir un posible solapamiento en el bloque de predicción. La intensidad del filtro de paso bajo puede variar según el valor de *angle.* Por ejemplo, cuando *angle* es igual a -*size*, puede aplicarse un filtrado de paso bajo débil y cuando *angle* es igual a -2, puede aplicarse un filtrado de paso bajo fuerte.

Tal como se ha explicado anteriormente, no todos los píxeles de referencia en *refV* se copian en *refH.* Dado que no se copian todos los píxeles de referencia en *refV*, se pierde algo de información cuando se copian los píxeles. Para mitigar la pérdida de información, puede duplicarse la resolución de píxeles de referencia en *refH* y *refV* de modo que *refH* y *refV* contienen no solo píxeles de bloques anteriormente codificados y reconstruidos, sino también un

píxel entre dos píxeles reconstruidos adyacentes que se genera interpolando dos píxeles adyacentes. Puede calcularse simplemente el promedio de dos píxeles adyacentes para generar un píxel de interpolación. El procedimiento de interpolación puede realizarse cuando se leen píxeles de referencia en la etapa 900 de la figura 9. Cuando se duplica la resolución de píxeles en *refH* y *refV,* se necesita ajustar a escala identificaciones de las direcciones de píxeles de referencia almacenados en *refH* y *refV*, tal como se realiza en las etapas 907, 910, 915 y 919 en la figura 9, y la etapa 1001 en la figura 10. Por ejemplo, se necesita cambiar *"int* + *col* + 1" realizado en las etapas 907, 910 y 915 por *"int* + 2 x *col* + 2". Se necesita cambiar *"int* + *col* + 2" realizado en las etapas 907, 910, 915 por "*int* + 2 x *col* + 3". Se necesita cambiar *"int2* + row + 1" e *"int2* + row + 2" realizados en la etapa 919 por *"int2* + 2 x row + 2" e *"int2* + 2 x row + 3", respectivamente.

En otro modo de realización, el procedimiento de la etapa 1202 en la figura 12 puede cambiarse simplemente por "*refH* [*col*]←*refV* [-*col*]" para simplificar adicionalmente el procedimiento de copiado. Aunque se degrada la exactitud de predicción, este modo de realización proporciona la menor complejidad a la operación de intra-predicción.

La figura 11B muestra la parte extendida 1104 añadida a *refH*. No se necesita que la parte extendida 1104 esté formada con píxeles de referencia de *refV*. La parte extendida 1104 puede formarse con píxeles de un área de bloque anteriormente reconstruido, que corresponde espacialmente a la ubicación de la parte extendida 1104. En la figura 11B, dado que se extiende en la dirección negativa, *refH* extendido (partes 1102 y 1104) oscila entre -*size* + 1 y 2x*size*. El intervalo de *refH* extendido puede volver a ajustarse a escala para oscilar entre 0 y 3x*size* - 1 añadiendo una compensación apropiada cuando se abordan píxeles de referencia en *refH* extendido. Lo mismo es cierto para volver a ajustar a escala el intervalo de *refV*.

En otro modo de realización, el límite de intervalo de *angle* puede elegirse libremente. En los modos de realización anteriores, se supone que *angle* adopta un valor dentro de un intervalo de desde -*size* hasta *size* (-*size* ≤ *angle* ≤ *size*). En otras palabras, en los modos de realización anteriores, los límites de intervalo de *angle* están definidos con el tamaño del bloque objetivo. Obsérvese que los límites de intervalo de *angle* pueden definirse independientemente del tamaño del bloque objetivo, aunque todavía es preferible que el límite de intervalo se defina con una potencia entera de 2, de manera que *log2_rangelimit* sea un número entero positivo y la ecuación *"rangelimit* = 1 << *log2_rangelimit"* siga siendo cierta. Al elegir un número grande adecuado para *rangelimit,* puede establecerse un gran número de direcciones de predicción y representarse mediante valores de *angle* a intervalos angulares lo suficientemente amplios.

Si el límite de intervalo de *angle* se define independientemente del tamaño del bloque objetivo, se necesita sustituir *size* que aparece en las figuras 9 y 10 por *rangelimit* y se necesita sustituir *log2_size* por *log2_rangelimit,* excepto para las etapas 909, 912, 917 y 921. También se necesita sustituir la comparación de "*angle* ≥ -1" realizada en la etapa 1000 de la figura 10 por *"angle* x *size*/*rangelimit*≥ -1" o *"angle* x *size* ≥ *-rangelimit".* Además, se necesita sustituir *size* que aparece en las etapas 1202 y 1301 en las figuras 12 y 13 por *rangelimit* y se necesita sustituir la comparación de "*¿col = angle?*" realizada en la etapa 1204 por *"¿col = angle x size*/*rangelimit?"*.

Si se introduce *rangelimit* como límite de intervalo de *angle,* la tabla 1 (proporcionada anteriormente) puede cambiarse de la siguiente forma:

Tabla 2

*angle**	2	5	9	13	17	21	26	32
InvAngle	4096	1638	910	630	482	390	315	256

En la tabla 2, se establece *rangelimit* a 32. *Angle** es igual a una aproximación de número entero de *"rangelimit* x *tan (*π x *angle*/*8)*", donde *angle* = 1, 2, 3, 4, 5, 6, 7 y 8. *InvAngle* es igual a *256* x *rangelimit*/*angle**. Los valores en la tabla 2 son todos números enteros que se obtienen mediante redondeo al alza. En lugar de redondearse al alza, los números pueden redondearse a la baja. En la tabla 3 proporcionada a continuación, *InvAngle* es igual a 32 x *rangelimitlangle**. Dado que se usa "32" en lugar de "256", la exactitud de predicción es necesariamente inferior a la de la tabla 2.

Tabla 3

*angle**	2	5	9	13	17	21	26	32
InvAngle	512	204	113	78	60	48	39	32

La figura 14 es un diagrama de flujo que muestra otro modo de realización que simplifica adicionalmente el procedimiento mostrado en la figura 10. El procedimiento mostrado en la figura 10 de copiar píxeles de referencia de *refV* en *refH* se realiza antes de que el flujo entre en el bucle de predicción principal, mientras que el procedimiento de copiado mostrado en la figura 14 se realiza dentro del bucle de predicción principal. Además, el procedimiento mostrado en la figura 14 elimina la variable *InvAngle.* Las etapas 900, 902 y 921 mostradas en la figura 14 son de las etapas correspondientes en la figura 10.

En la etapa 1401, se inicia un contador *lastInt* a -1. *lastInt* representa el índice del último píxel que se añadió a *refH.* En la etapa 902, se calcula *pos* mediante *angle* x *(row + 1)*. Tal como se ha explicado anteriormente, *pos* identifica la posición de una intersección entre los límites y la dirección de predicción representada por *angle.* En el contexto de la figura 9, la etapa 902 produce *pos*, que identifica la posición de una intersección entre el límite horizontal y la dirección de predicción representada por *angle.* Además, en la etapa 902, una parte entera en *pos* se almacena en *int* y una parte fraccionaria en *pos* se almacena en un parámetro "*frac*". En la etapa 1402, se determina si *int* es inferior a *lastInt*. Si *int* es inferior a *lastInt*, un píxel de referencia en *refV* identificado mediante *row* se copia en *refH* en una dirección identificada mediante "*int* + 1". La etapa 1404 consiste en las etapas 904, 905, 906, 907, 908, 909, 910, 911 y 912 mostradas en las figuras 9 y 10, cuya descripción no se repite aquí. En la etapa 1405, *int* se copia en *lastInt*. La operación de copiar *int* en *lastInt* puede realizarse en la etapa 1403, en lugar de la etapa 1405.

La operación de copiado en la etapa 1403 da como resultado copiar el mismo píxel que se copió en las etapas 1202 y 1302, en las que se usa redondeo a la baja en esas etapas. La etapa 1403 puede modificarse para redondear al número entero más próximo usando de manera condicional "*row* + 1", en lugar de "row", en la etapa 1403 cuando la posición fraccionaria *frac* calculada en la etapa 902 es mayor que *offset,* lo cual se identifica mediante *rangelimit* + (*angle* >> 1). Obsérvese que *angle* es negativo y *frac* es positivo. El uso de "*row* + 1" da como resultado redondeo al alza. Para realizar el incremento condicional de *row* en 1, se cambia el procedimiento realizado en la etapa 1403 por *refH*[*int +1*] ← *refV*[*row* - ((*offset* - *frac*) » 31)]; suponiendo que en una aritmética de 32 bits, el desplazamiento a la derecha de *"offset - frac"* da como resultado -1 cuando *frac* es mayor que *offset* y da como resultado 0 en caso contrario. Por lo tanto, el identificador de dirección *"row* - ((*offset - frac*) >> 31)" se convierte en "*row* + 1" cuando *frac* es mayor que *offset* y se convierte en "*row*" en caso contrario. Si se establece *offset* a *rangelimit*, "*offset-frac*" será siempre positivo y, por lo tanto, no se producirá ningún redondeo.

A continuación se enumera el código fuente desarrollado en el lenguaje de programación C++, que implementa el procedimiento mostrado en la figura 14. El código fuente se modifica de la función TComPredictiopn::xPredIntraAng encontrada en el archivo TComPrediction.cpp que es parte del software TMuC 0.7 desarrollado por JCT-VC, que está disponible en http://hevc.kw.bbc.co.uk/svn/jctvc.a124/tags/0.7.

```
// Función para obtener las intra-predicciones angulares simplificadas

Void TComPrediction::xPredIntraAng (Int* pSrc, Int iSrcStride, Pel*& rpDst, Int iDstStride, UInt iWidth, UInt iHeight, UInt uiDirMode, Bool bAbove, Bool bLeft) {

  Int k, l;

  Int deltaInt, deltaFract, refMainIndex;

  Int intraPredAngle = 0;

  Int absAng = 0;

  Int signAng = 0;

  Int blkSize = iWidth;

  Bool modeDC = false;

  Bool modeVer = false;

  Bool modeHor = false;

  Pel* pDst = rpDst;

// Mapear el índice de modo a la dirección de predicción principal y el ángulo

  if (uiDirMode == 0)

    modeDC = true;

  else if (uiDirMode < 18)

    modeVer = true;

  else
```

```
  modeHor = true;
intraPredAngle = modeVer ? uiDirMode - 9 : modeHor ? uiDirMode - 25 : 0;
absAng = abs(intrapredAngle);
signAng = intraPredAngle < 0 ? -1 : 1;
// Establecer desplazamientos de bits y ajustar a escala el parámetro de ángulo a size2
Int iAngTable[9] = { 0, 2, 5, 9, 13, 17, 21, 26, 32};
absAng = iAngTable[absAng];
intraPredAngle = signAng * absAng;
// Realizar la predicción DC
if (modeDC) {
  Pel dcval = predIntraGetPredValDC(pSrc, iSrcStride, iWidth, iHeight, bAbove, bLeft);
    for (k=0;k<blkSize;k++) {
      for (l=0;l<blkSize;1++) {
        pDst(k*iDstStride+1] = dcval;
      }
    }
  }
  // Realizar predicciones angulares
  else {
    Pel tmp;
    Int *pSrcTL = pSrc - iSrcStride - 1;
    Int iStepMain = (modeVer) ? 1 : iSrcStride;
    if (intraPredAngle == 0) {
      for (k=0;k<blkSize;k++) {
        for (l=0;l<blkSize;1++) {
          pDst [k*iDstStride+1] = pSrcTL[(1+1) * iStepMain];
        }
      }
    }
    else {
      Int iStepSide = (modeVer) ? iSrcStride 1;
      int lastDeltaInt = -1;
      Int iOffset = 32 + (intraPredAngle >> 1); // permite redondear a la referencia lateral más próxima
```

```
// Int iOffset = 32;                    // sin redondeo.
  Pel ref [2*MAX_CU_SIZE];
  Pel* refMain = ref + ((intraPredAngle < 0) ? blkSize : 0);
  if (intraPredAngle > 0) {
    for (k = 0; k < 2*blkSize; k++)
      refMain[k] = pSrcTL[(k+1) * iStepMain];
  }
  else {
    for (k = -1; k < blkSize; k++) // el resto se copia más tarde en la etapa 1403, según y cuando se requiera
    refMain[k] = pSrcTL[(k+1) * iStepMain];
  }
  for (k = 0; k < blkSize; k++) {
  Int deltaPos = (k+1) * intraPredAngle;
  deltaInt = deltaPos >> 5;
  deltaFract = deltaPos & (32 - 1);
  if (deltaInt < lastDeltaInt) { // etapa 1402
    lastDeltaInt = deltaInt;
    refMain[deltaInt] = pSrcTL[(k-((iOffset-deltaFract)>>31))*iStepSide]; // etapa 1403
  }
  // etapa 1404
  if (deltaFract) {
    // Realizar filtrado lineal
    for (l=0;l<blkSize;1++) {
      refMainIndex = 1+deltaInt;
      pDst[k*iDstStride+1] = (Pel) (((32-deltaFract) * refMain[refMainIndex] + deltaFract *
refMain[refMainIndex+1] + 16) >> 5 );
    }
  }
  else {
    // Simplemente copiar las muestras de números enteros
    for (l=0;l<<blkSize;l++) {
      pDst[k*iDstStride+1] = refMain[1+deltaInt];
    }
  }
```

```
    }
  }
  // Dar la vuelta al bloque si esto es el modo horizontal
  if (modeHor) {
    for (k=0;k<blkSize-1;k++) {
      for (l=k+1;l<blkSize;1++) {
        tmp = pDst[k*iDstStride+1];
          pDst(k*iDstStride+1] = pDst(1*iDstStride+k];
          pDst[1*iDstStride+k] = tmp;
        }
      }
    }
  }
```

Aunque sin duda se le ocurrirán muchas alteraciones y modificaciones de la presente invención a un experto habitual en la técnica, tras haber leído la descripción anterior debe entenderse que no se pretende de ninguna manera que ningún modo de realización particular mostrado y descrito a modo de ilustración se considere limitativo. Por tanto, no se pretende que referencias a detalles de diversos modos de realización limiten el alcance de las reivindicaciones, que en sí mismas solo mencionan aquellas características que se consideran esenciales para la invención.

REIVINDICACIONES

1. Un método de descodificación de vídeo que comprende etapas ejecutables por ordenador ejecutadas por un procesador de un codificador de vídeo para implementar:

obtener un valor de un parámetro de ángulo inverso (*InvAngle*) de una tabla de consulta que indica valores de parámetro de ángulo inverso (*InvAngle*) en relación con los valores de un parámetro de ángulo (*angle**) que representan una dirección de predicción;

identificar al menos algunos píxeles entre píxeles de límite vertical, mediante el uso de un identificador de píxeles verticales que se expresa mediante una función usando *[col x InvAngle]*, donde *col* es un contador que se reduce en 1 desde -1 hasta (*angle** x *size* / *rangelimit*), donde *size* es un tamaño de un bloque objetivo y *rangelimit* define un intervalo del parámetro de ángulo (*angle**);

recuperar los al menos algunos píxeles, según una dirección de predicción de intra-predicción en un bloque objetivo que va a predecirse, de una primera área de memoria (*refV*) en la que se almacena una matriz de píxeles de límite vertical, en el que los píxeles de límite vertical se encuentran directamente a la izquierda del bloque objetivo;

añadir los píxeles recuperados a una matriz de píxeles de límite horizontal que se encuentra directamente por encima del bloque objetivo, en el que los píxeles recuperados se añaden directamente al extremo izquierdo de la matriz de píxeles de límite horizontal para formar una secuencia consecutiva de los píxeles de límite horizontal;

almacenar los píxeles añadidos en una segunda área de memoria (*refH*) en la que se almacena la matriz de píxeles de límite horizontal, para extender la matriz almacenada en la segunda área de memoria (*refH*) de la misma; y

realizar la intra-predicción del bloque objetivo usando solo los píxeles de límite horizontal incluyendo los píxeles añadidos, de la matriz extendida almacenada en la segunda área de memoria (*refH*) como píxeles de referencia,

en el que la identificación de los al menos algunos píxeles entre los píxeles de límite vertical comprende un desplazamiento aritmético a la derecha en el que (*col* x *InvAngle* + 128) se desplaza a la derecha en 8 dígitos binarios.

2. Un descodificador de vídeo que comprende un procesador de un sistema informático y una memoria que almacena programas ejecutables por el procesador para:

obtener un valor de un parámetro de ángulo inverso (*InvAngle*) de una tabla de consulta que indica valores de parámetro de ángulo inverso (*InvAngle*) en relación con los valores de un parámetro de ángulo (*angle**) que representan una dirección de predicción;

identificar al menos algunos píxeles entre píxeles de límite vertical, mediante el uso de un identificador de píxeles verticales que se expresa mediante una función usando *[col x InvAngle]*, donde *col* es un contador que se reduce en 1 desde -1 hasta (*angle** x *size* / *rangelimit*), donde *size* es un tamaño de un bloque objetivo y *rangelimit* define un intervalo del parámetro de ángulo (*angle**);

recuperar los al menos algunos píxeles, según una dirección de predicción de intra-predicción en un bloque objetivo que va a predecirse, de una primera área de memoria (*refV*) en la que se almacena una matriz de píxeles de límite vertical, en el que los píxeles de límite vertical se encuentran directamente a la izquierda del bloque objetivo;

añadir los píxeles recuperados a una matriz de píxeles de límite horizontal que se encuentra directamente por encima del bloque objetivo, en el que los píxeles recuperados se añaden directamente al extremo izquierdo de la matriz de píxeles de límite horizontal para formar una secuencia consecutiva de los píxeles de límite horizontal;

almacenar los píxeles añadidos en una segunda área de memoria (*refH*) en la que se almacena la matriz de píxeles de límite horizontal, para extender la matriz almacenada en la segunda área de memoria (*refH*) de la misma; y

realizar la intra-predicción del bloque objetivo usando solo los píxeles de límite horizontal incluyendo los píxeles añadidos, de la matriz extendida almacenada en la segunda área de memoria (*refH*) como píxeles de referencia,

en el que la identificación de los al menos algunos píxeles entre los píxeles de límite vertical comprende un desplazamiento aritmético a la derecha en el que (*col* x *InvAngle* + 128) se desplaza a la derecha en 8 dígitos binarios.

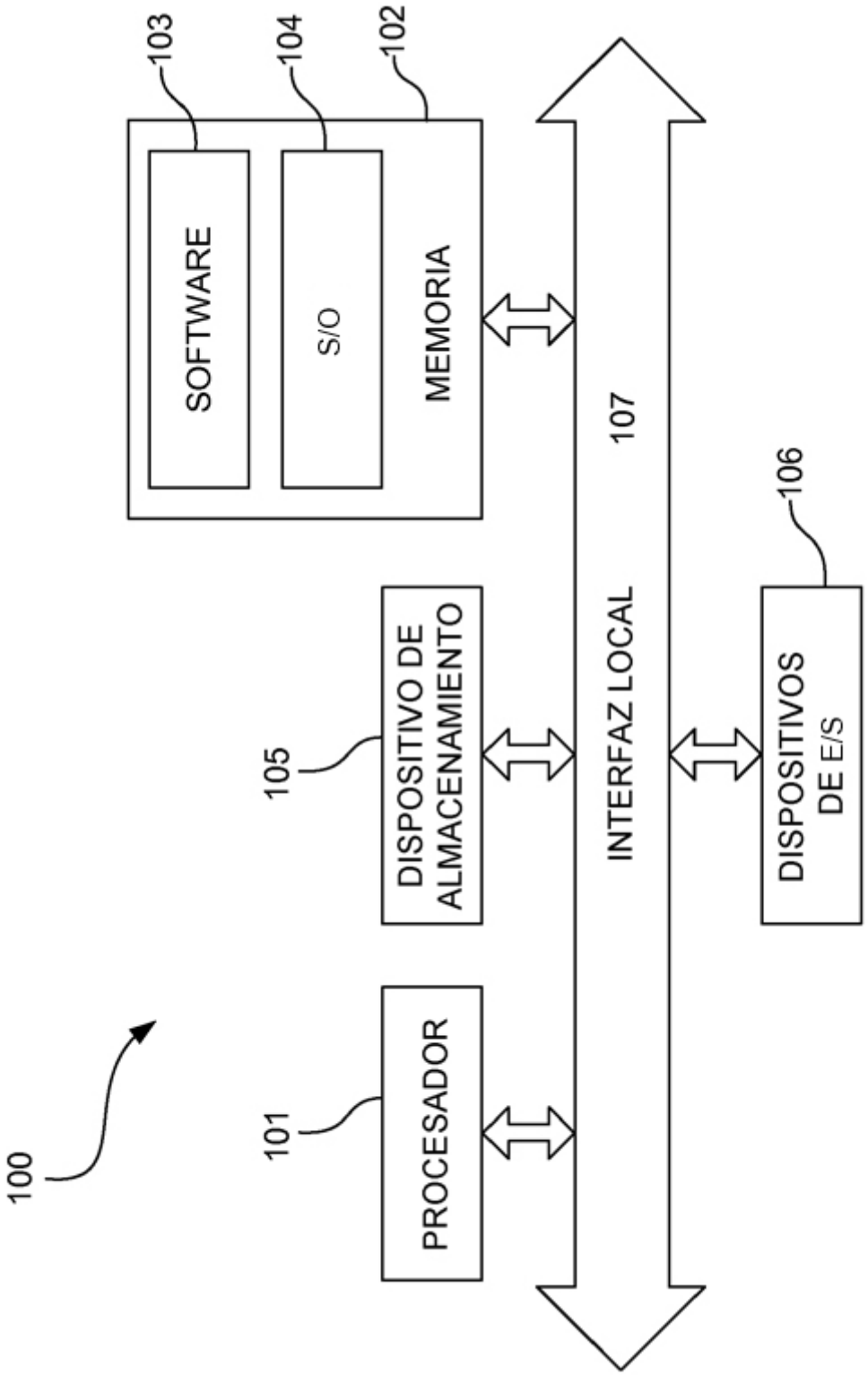
100
101
PROCESADOR
105
DISPOSITIVO DE ALMACENAMIENTO
INTERFAZ LOCAL
107
106
DISPOSITIVOS DE E/S
103
SOFTWARE
104
S/O
102
MEMORIA

Fig. 1

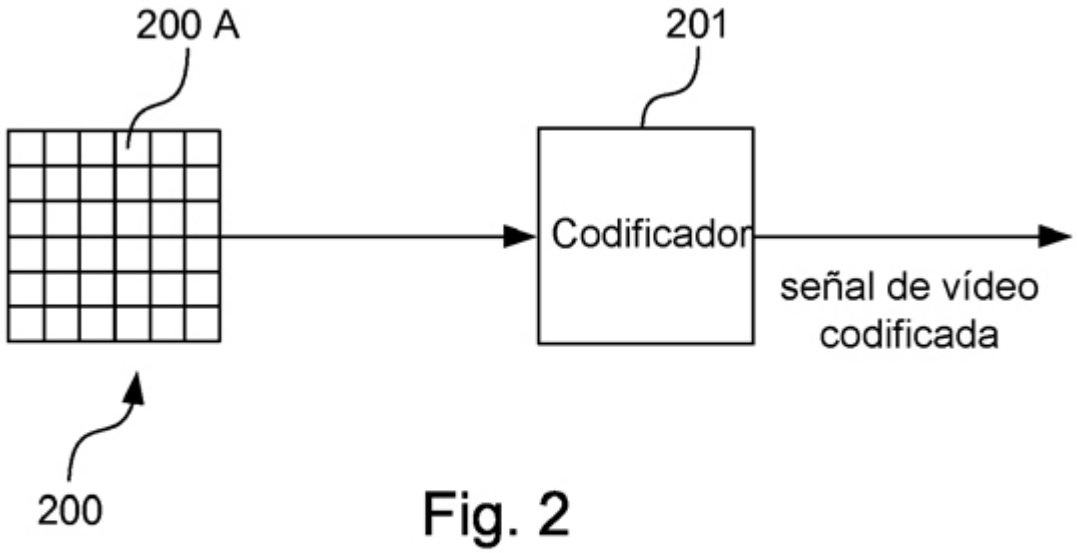
200 A
201
Codificador
señal de vídeo codificada
200

Fig. 2

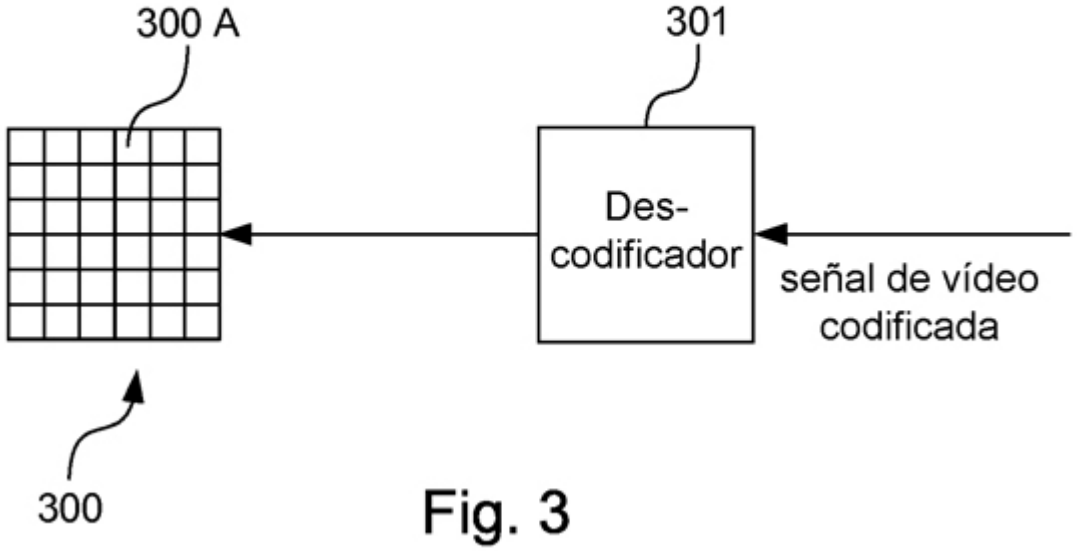
300 A
301
Des-codificador
señal de vídeo codificada
300

Fig. 3

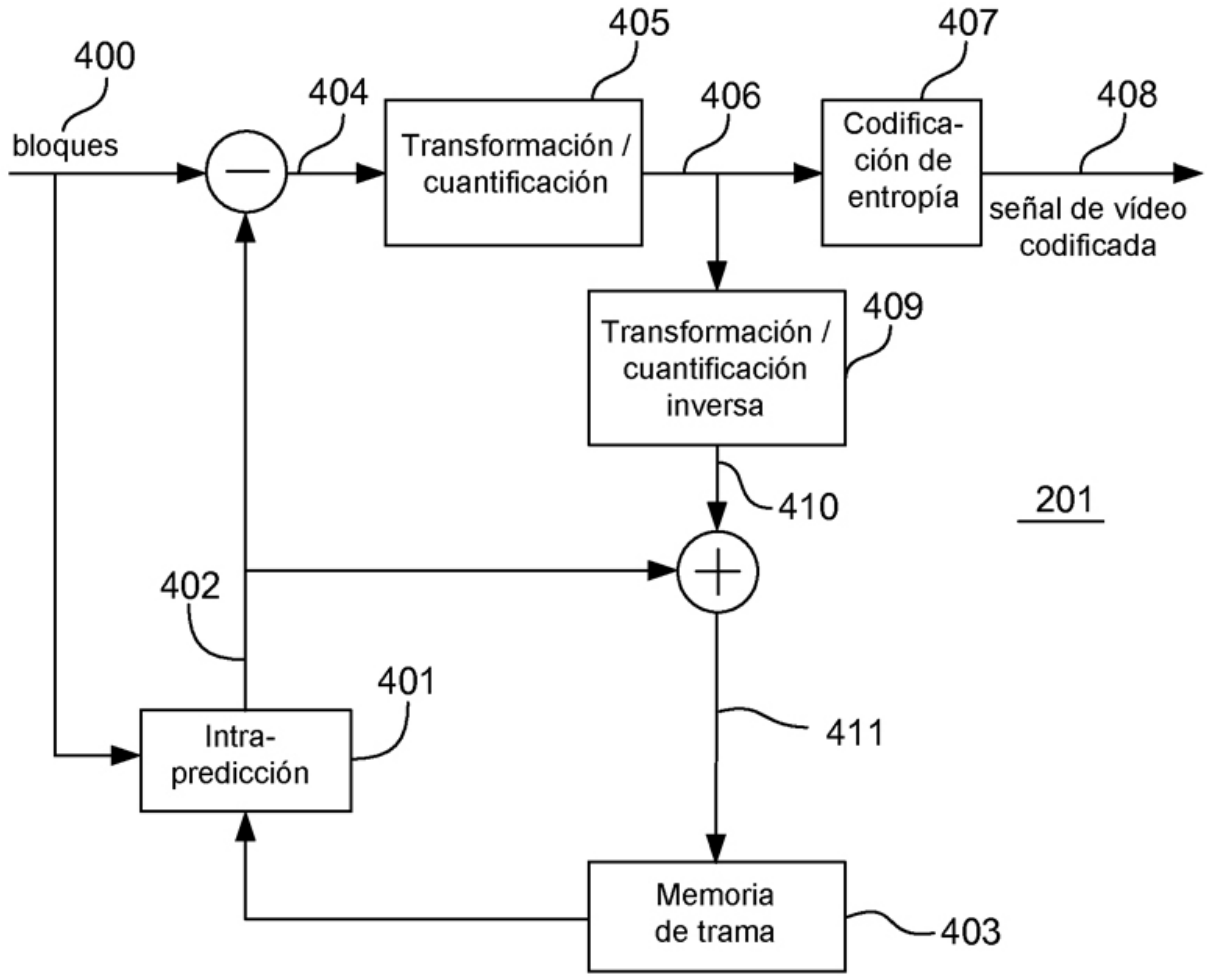
400
bloques
404
405
Transformación / cuantificación
406
407
Codifica-ción de entropía
408
señal de vídeo codificada
409
Transformación / cuantificación inversa
410
201
402
401
Intra-predicción
411
Memoria de trama
403

Fig. 4

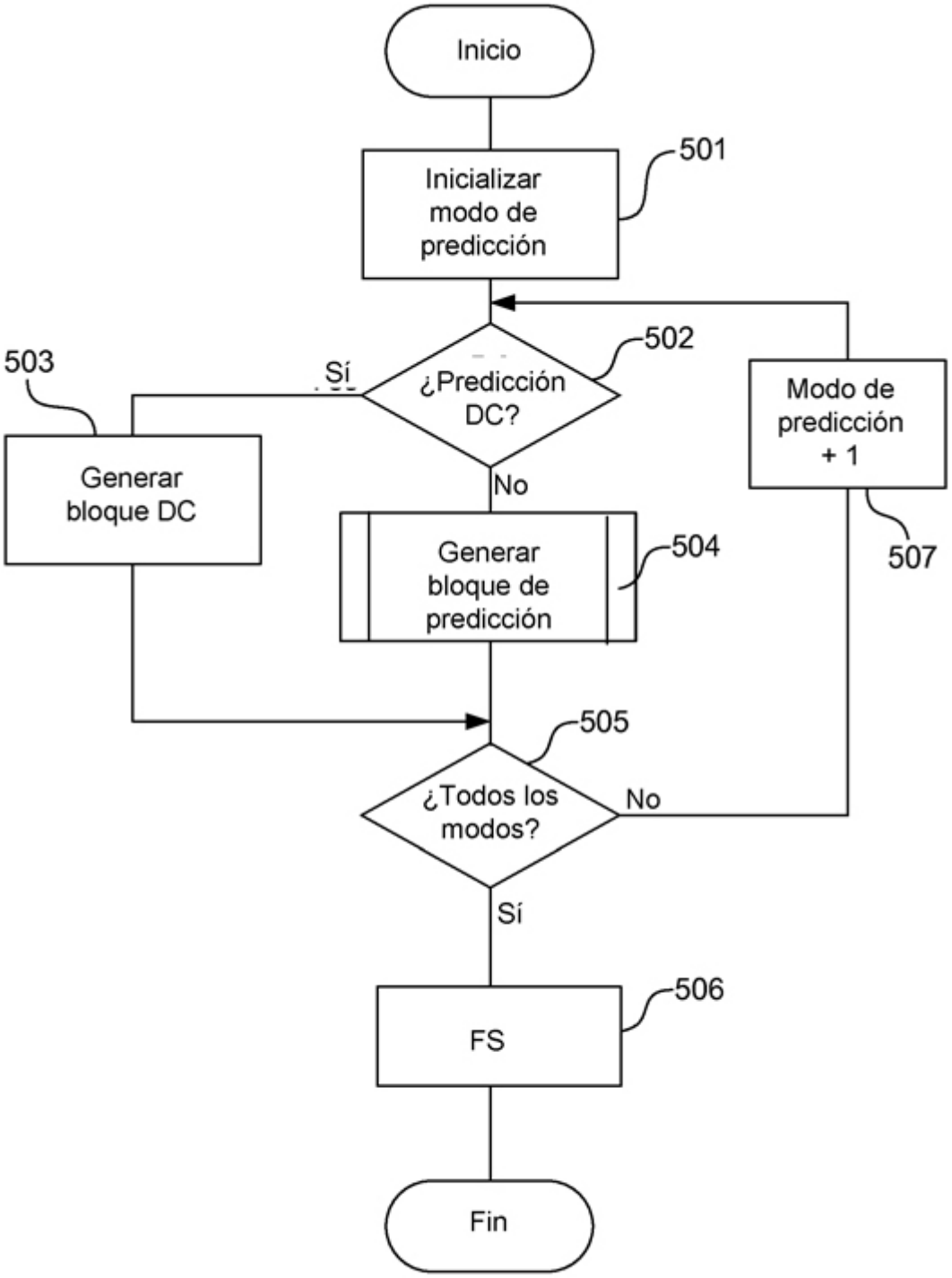
Inicio
Inicializar modo de predicción
501
¿Predicción DC?
502
Sí
503
Generar bloque DC
No
Generar bloque de predicción
504
Modo de predicción + 1
507
¿Todos los modos?
505
No
Sí
FS
506
Fin

Fig. 5

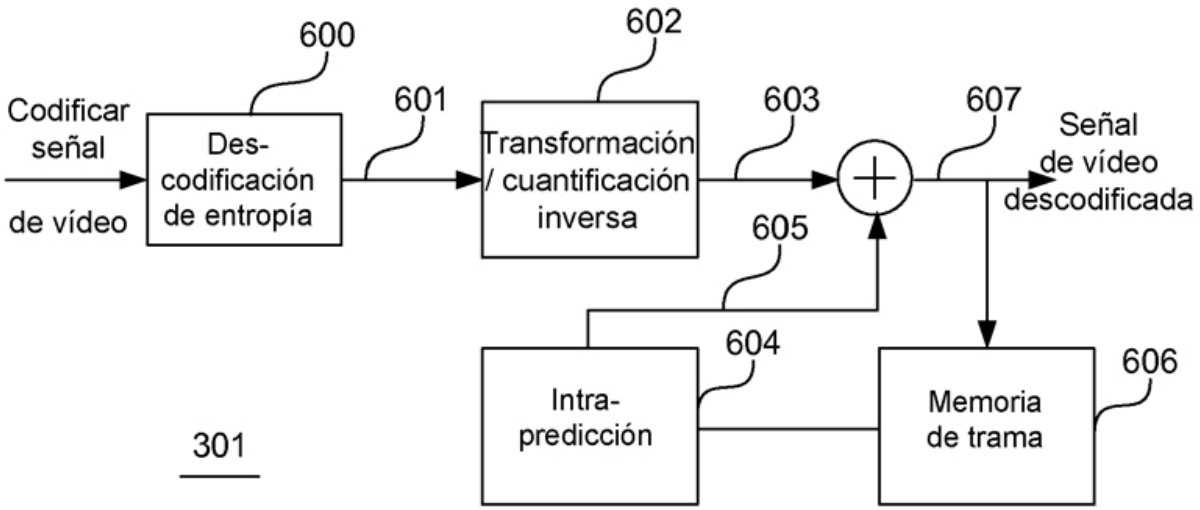
600
602
Codificar
señal
de vídeo
Des-
codificación
de entropía
601
Transformación
/ cuantificación
inversa
603
607
Señal
de vídeo
descodificada
605
604
606
Intra-
predicción
Memoria
de trama
301

Fig. 6

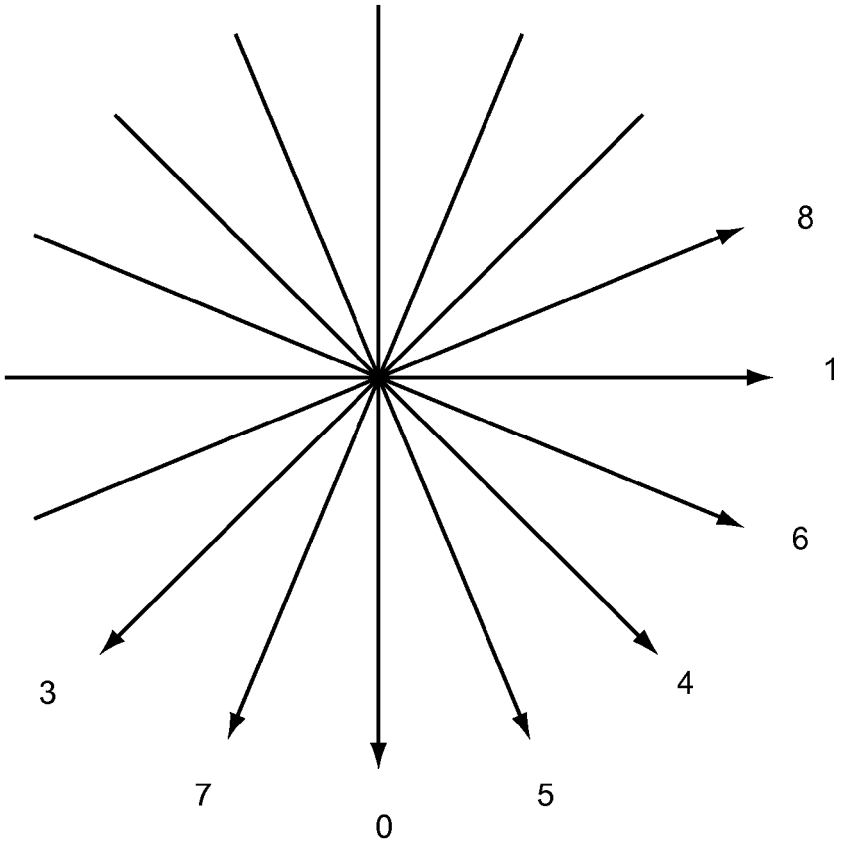

Fig. 7

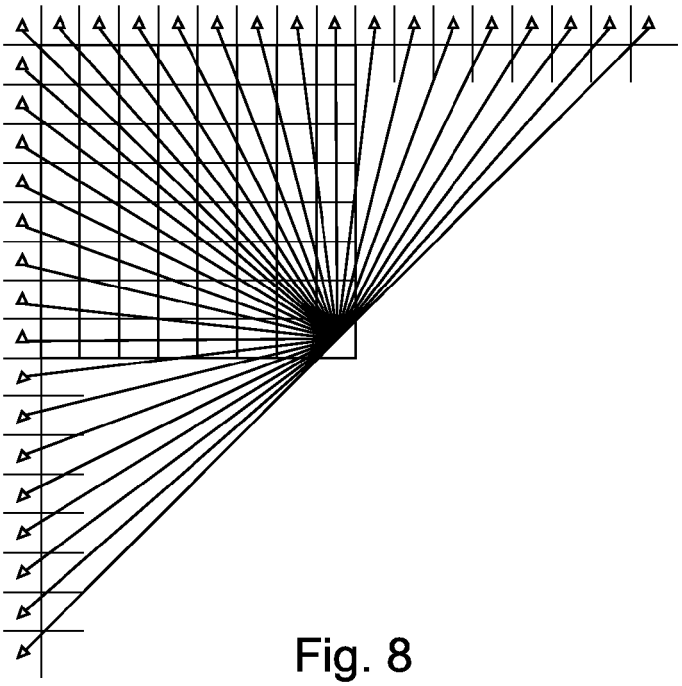

Fig. 8

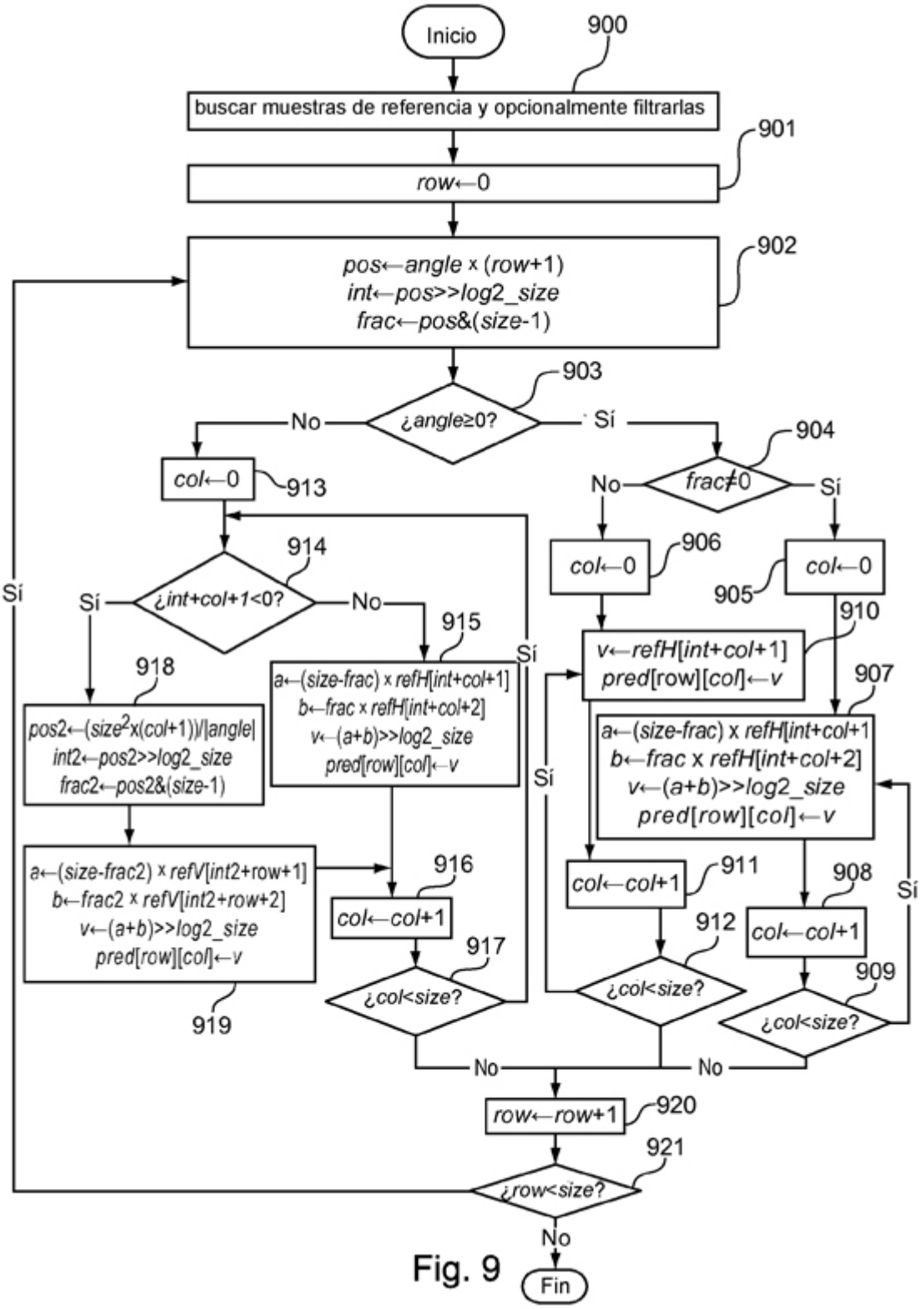
Inicio
900
buscar muestras de referencia y opcionalmente filtrarlas
901
row←0
902
pos←angle x (row+1)
int←pos>>log2_size
frac←pos&(size-1)
903
¿angle≥0?
No
Sí
904
frac≠0
No
Sí
col←0
913
col←0
906
col←0
905
914
¿int+col+1<0?
Sí
No
Sí
915
910
v←refH[int+col+1]
pred[row][col]←v
907
a←(size-frac) x refH[int+col+1]
b←frac x refH[int+col+2]
v←(a+b)>>log2_size
pred[row][col]←v
Sí
918
pos2←(size2x(col+1))/|angle|
int2←pos2>>log2_size
frac2←pos2&(size-1)
a←(size-frac2) x refV[int2+row+1]
b←frac2 x refV[int2+row+2]
v←(a+b)>>log2_size
pred[row][col]←v
919
916
col←col+1
917
¿col<size?
911
col←col+1
912
¿col<size?
908
col←col+1
909
¿col<size?
Sí
No
No
row←row+1
920
921
¿row<size?
Sí
No
Fin

Fig. 9

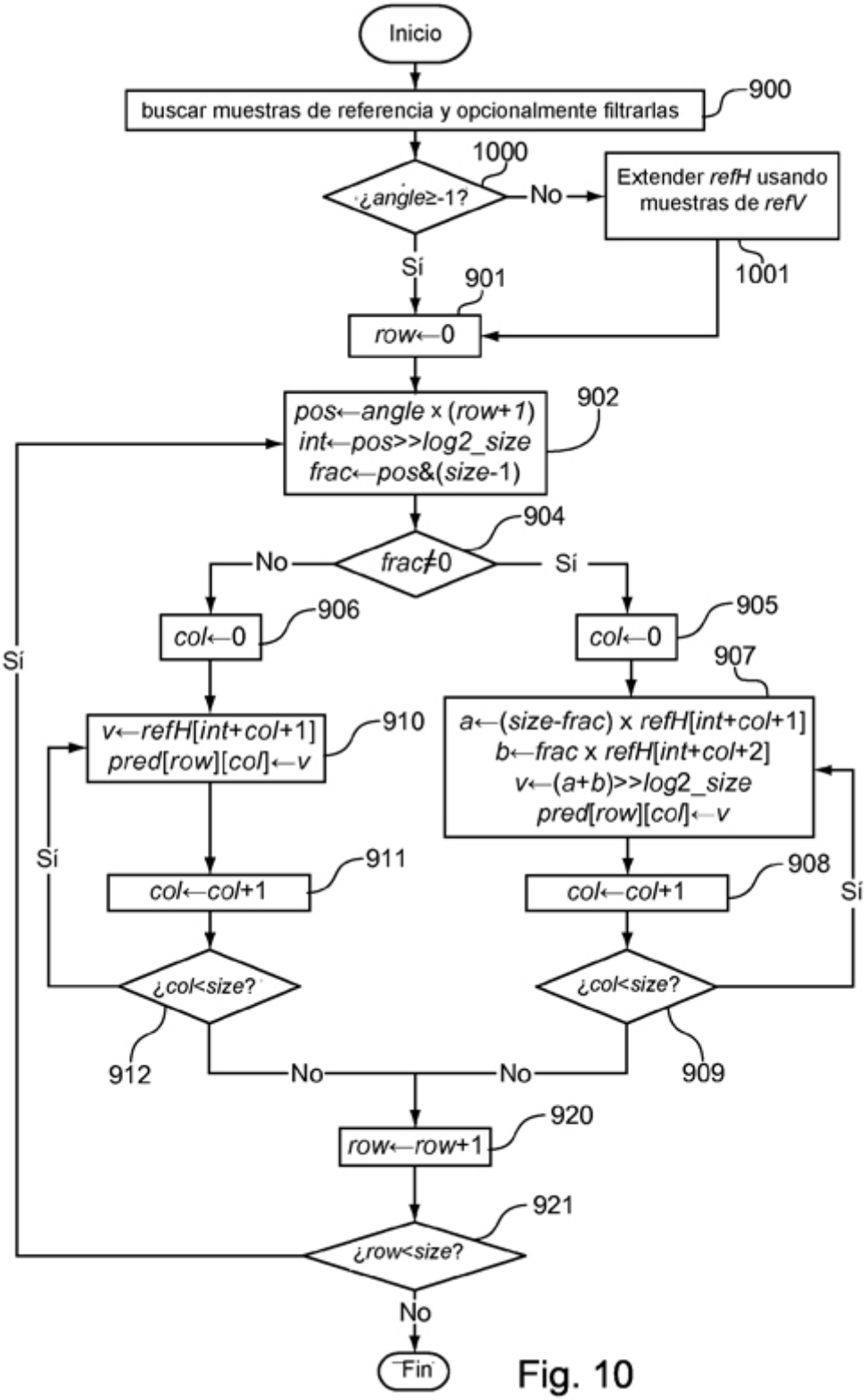
Inicio
buscar muestras de referencia y opcionalmente filtrarlas
900
1000
¿angle≥-1?
No
Extender refH usando muestras de refV
1001
Sí
901
row←0
902
pos←angle x (row+1)
int←pos>>log2_size
frac←pos&(size-1)
904
frac≠0
No
Sí
906
col←0
905
col←0
907
Sí
v←refH[int+col+1]
pred[row][col]←v
910
a←(size-frac) x refH[int+col+1]
b←frac x refH[int+col+2]
v←(a+b)>>log2_size
pred[row][col]←v
Sí
911
col←col+1
908
col←col+1
Sí
¿col<size?
¿col<size?
912
No
No
909
920
row←row+1
921
¿row<size?
No
Fin

Fig. 10

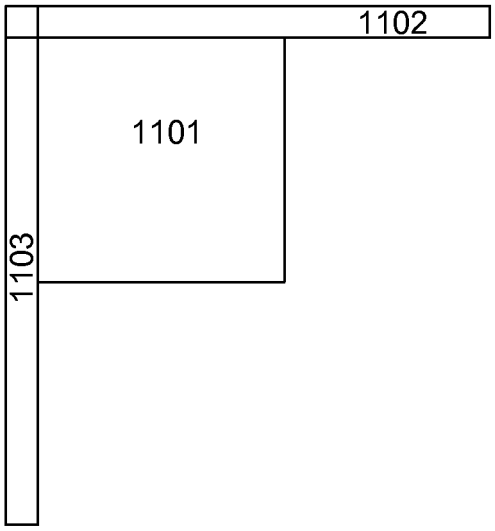

Fig. 11A

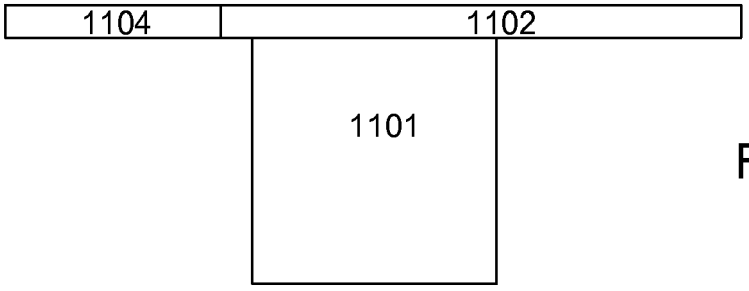

Fig. 11B

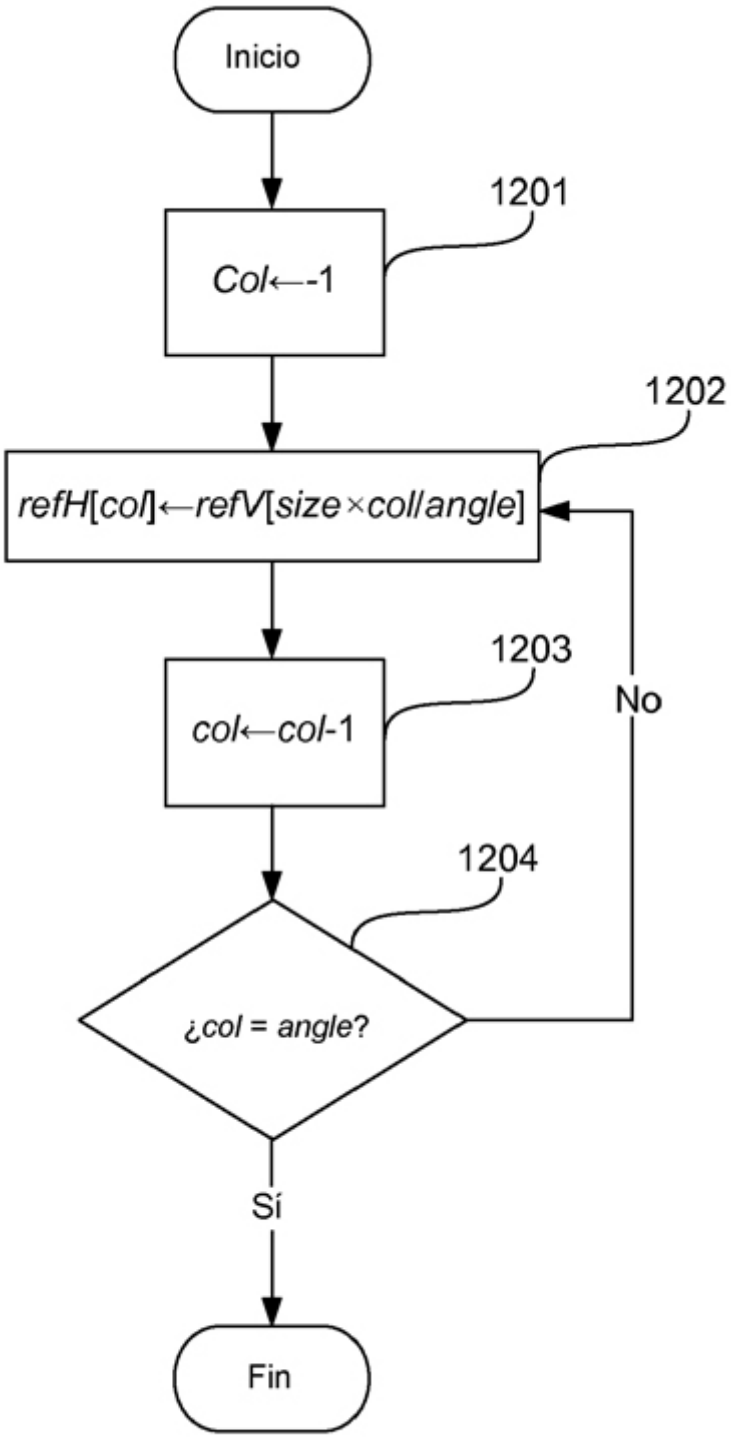
Inicio
1201
Col←-1
1202
refH[col]←refV[size×col/angle]
1203
col←col-1
No
1204
¿col = angle?
Sí
Fin

Fig. 12

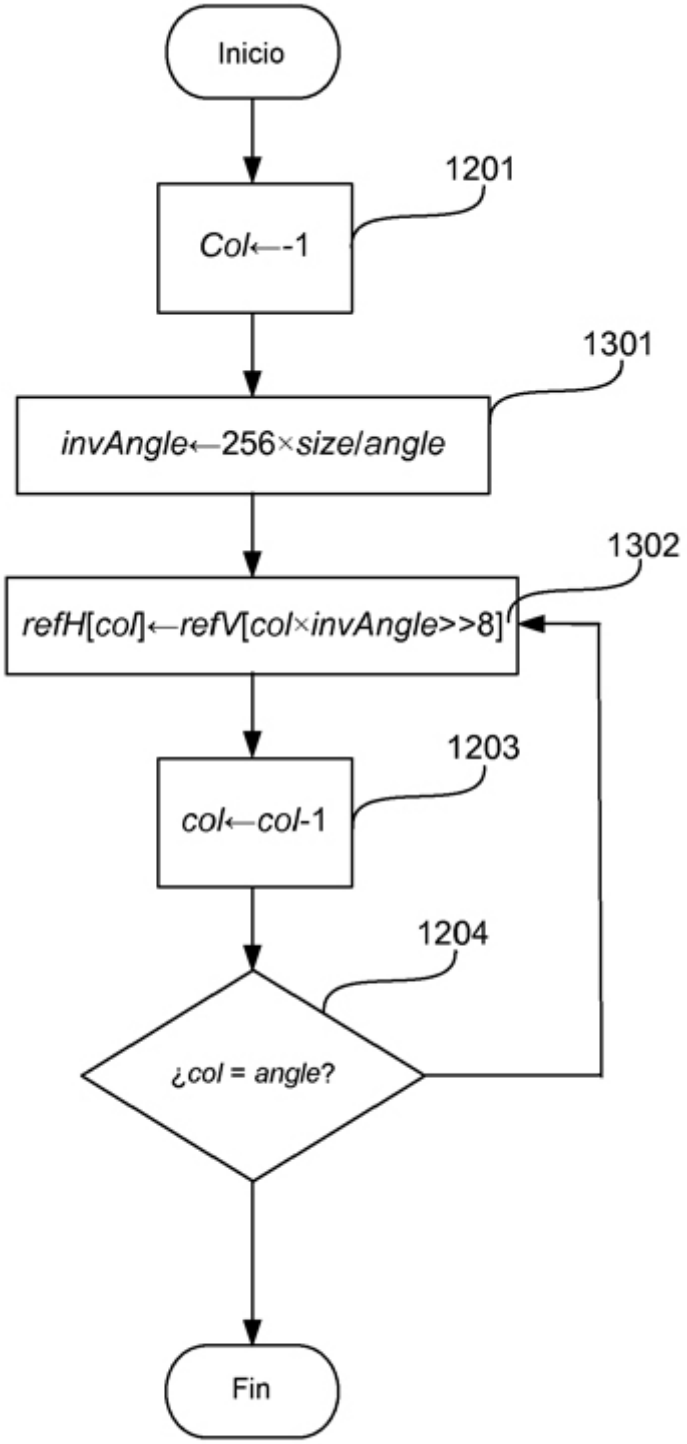
Inicio
1201
Col←-1
1301
invAngle←256×size/angle
1302
refH[col]←refV[col×invAngle>>8]
1203
col←col-1
1204
¿col = angle?
Fin

Fig. 13

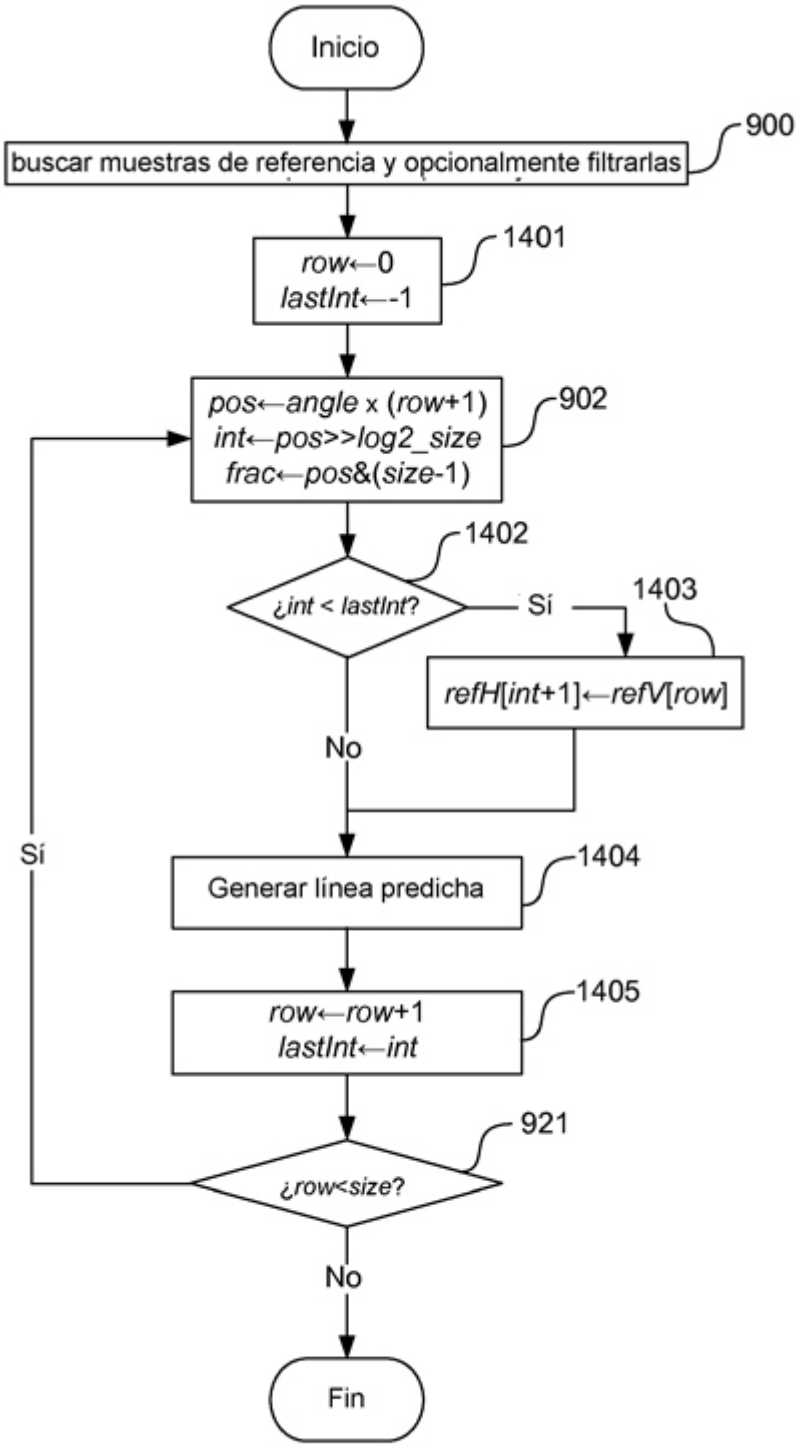
Inicio
buscar muestras de referencia y opcionalmente filtrarlas
900
row←0
lastInt←-1
1401
pos←angle x (row+1)
int←pos>>log2_size
frac←pos&(size-1)
902
¿int < lastInt?
1402
Sí
1403
refH[int+1]←refV[row]
No
Sí
Generar línea predicha
1404
row←row+1
lastInt←int
1405
¿row<size?
921
No
Fin

Fig. 14