(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 611 160**

(51) Int. CI.:

H04N 19/40 (2014.01)
H04N 19/46 (2014.01)
H04N 19/48 (2014.01)
H04N 19/513 (2014.01)
H04N 19/70 (2014.01)
H04N 19/85 (2014.01)
H04N 19/88 (2014.01)
H04N 21/426 (2011.01)
H04N 21/4405 (2011.01)
H04N 21/4408 (2011.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **11.12.2013 E 13290312 (1)**

(97) Fecha y número de publicación de la concesión europea: **19.10.2016 EP 2884748**

(54) Título: **Aparato y método para decodificar vídeo comprimido**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
05.05.2017

(73) Titular/es:

SQUADEO S.A.S. (100.0%)
105 Avenue de l'Europe
78160 Marly le Roi , FR

(72) Inventor/es:

LIU, XIAOBO y
MARTIN, FRANÇOIS

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Aparato y método para decodificar vídeo comprimido

Esta invención se refiere a la decodificación de vídeo digital comprimido. Es particularmente pertinente a una aplicación de software para reproducir vídeo, que usa un dispositivo decodificador de hardware especializado para realizar la decodificación (descompresión).

La distribución de vídeo a través de internet es un mercado que crece rápidamente. El denominado contenido "superpuesto" (Over The Top) (OTT) puede entregarse a través de una conexión de internet de banda ancha. Típicamente, este contenido se suministra mediante una tercera parte, siendo el Proveedor de Servicio de Internet (ISP) responsable únicamente de transportar los paquetes del Protocolo de Internet (IP) subyacentes. Los servicios OTT, en particular, representan un mercado de rápido crecimiento. Posibilitan a los usuarios finales disfrutar de vídeo en dispositivos inteligentes tales como teléfonos y tabletas inteligentes. Recientemente, se han hecho disponibles muchas aplicaciones de software, para ver vídeo sobre servicios OTT. Sin embargo la mayoría del contenido para estos servicios está limitado en calidad y resolución - es típicamente Definición Estándar (SD) en lugar de Alta Definición (HD).

Muchos dispositivos inteligentes están basados en sistemas operativos abiertos (tales como Android). Esto posibilita que los desarrolladores creen rápidamente aplicaciones para los dispositivos. Sin embargo, la naturaleza abierta de estos sistemas operativos puede ser de interés para los proveedores de contenido, que pueden ser reacios a proporcionar contenido de alta calidad a alta resolución en tales plataformas sin medidas apropiadas para proteger el contenido de copia no autorizada. Si pudiera proporcionarse protección de contenido adecuada, los proveedores de contenido podrían estar más dispuestos a distribuir contenido HD de alta calidad.

Una solución que se ha sugerido es usar un "entorno confiable", para proteger la ruta de entrega de vídeo completa. Un ejemplo de esto es la tecnología "TrustZone®" desarrollada por ARM Ltd. Una solución de este tipo puede resolver el problema de protección de contenido, pero puede ser complicada de implementar. En particular, la necesidad de certificados y otros mecanismos de seguridad fuertes en un entorno de este tipo pone una carga adicional en los desarrolladores de aplicación de software, que pueden necesitar pasar a través de un proceso de certificación cada vez que se actualiza la aplicación de software.

El documento de la técnica anterior SANGGU KWON et al: "Digital Video Scrambling Using Motion Vector and Slice Relocation", 1 de enero de 2005 (01-01-2005), Image Analysis and Recognition Lecture Notes in Computer Science; LNCS, Springer, Berlín, DE, páginas 207 - 214, XP019020136, ISBN: 978-3-540-29069-8 desvela aleatorización de un flujo de bits de vídeo usando una técnica de aleatorización que distorsiona deliberadamente las secuencias de vídeo original de una manera reversible relocalizando arbitrariamente los vectores de movimiento diferenciales y las posiciones de inicio de macrobloque en un corte. Esta técnica de aleatorización tiene por objeto superar el problema de que las técnicas de aleatorización de vídeo a menudo aumentan la tasa de bits de los datos de origen.

La invención se expone en el conjunto adjunto de las reivindicaciones.

En este aparato, el vídeo se decodifica (descomprime) en forma aleatorizada. Esto puede hacer más difícil para un usuario malicioso copiar el vídeo, puesto que el vídeo está en forma aleatorizada tanto en la entrada como en la salida del decodificador de vídeo.

La unidad de aleatorización de vídeo está adaptada preferentemente para aleatorizar el vídeo comprimido de tal manera que: no evita la decodificación del vídeo mediante el decodificador de vídeo; modifica el contenido visual del vídeo descomprimido aleatorizado después de decodificación; y es reversible mediante la unidad de desaleatorización de vídeo, de manera que el vídeo descomprimido desaleatorizado es idéntico o sustancialmente idéntico al vídeo que se produciría si el vídeo comprimido se hubiera de decodificar mediante el decodificador de vídeo, sin aleatorización.

Preferentemente la modificación del contenido visual crea una perturbación visual perceptible. Esto resultaría molesto para un observador que ve el vídeo descomprimido aleatorizado, haciendo de esta manera más difícil para un usuario malicioso hacer copias ilícitas de alta calidad del vídeo.

La aleatorización es sistemática en el sentido que es posible que la unidad de desaleatorización de vídeo reconstruya (de manera precisa o al menos aproximadamente) el vídeo descomprimido que se hubiera producido por el decodificador de vídeo si no se hubiera aplicado aleatorización.

Preferentemente, no se requiere modificación del decodificador de vídeo. Esto significa que el vídeo comprimido y el vídeo comprimido aleatorizado pueden decodificarse ambos mediante el decodificador de vídeo de la misma manera. Es decir, la aleatorización es transparente para el decodificador de vídeo.

El vídeo descomprimido desaleatorizado puede considerarse sustancialmente idéntico al vídeo descomprimido correspondiente sin aleatorización si contiene el mismo número de fotogramas y una mayoría de estos fotogramas son idénticos a sus homólogos en el vídeo descomprimido al que no se ha aplicado aleatorización. Preferentemente, cualquier fotograma que no sea idéntico a sus homólogos es visualmente similar a ellos.

Preferentemente, la unidad de aleatorización de vídeo está adaptada para producir metadatos de aleatorización que describen una operación de aleatorización realizada mediante la unidad de aleatorización de vídeo; y la unidad de desaleatorización de vídeo está dispuesta para recibir los metadatos de aleatorización desde la unidad de aleatorización y está adaptada para usarlos para desaleatorizar el vídeo descomprimido aleatorizado recibido desde el decodificador de vídeo.

Los metadatos de aleatorización preferentemente no se proporcionan al decodificador de vídeo.

La aleatorización realizada mediante la unidad de aleatorización de vídeo puede comprender una o más operaciones realizadas en unidades de acceso individuales del vídeo comprimido.

Es decir, la modificación no trata todas las unidades de acceso de manera uniforme. Al menos algunas unidades de acceso se tratan de manera diferente - una operación que se realiza en una unidad de acceso es diferente de una operación que se realiza en al menos otra unidad de acceso. Una unidad de acceso puede comprender la representación codificada de un único fotograma del vídeo o parte de un fotograma.

La aleatorización realizada mediante la unidad de aleatorización de vídeo puede comprender al menos uno de: cambiar una ordenación de unidades de acceso en el vídeo comprimido; insertar una o más unidades de acceso adicionales en el vídeo comprimido; y eliminar una o más unidades de acceso a partir del vídeo comprimido.

Preferentemente, cambiar la ordenación de unidades de acceso comprende cambiar su orden de presentación sin cambiar el orden en el que se decodifican mediante el decodificador de vídeo.

Opcionalmente, la unidad de aleatorización de vídeo puede adaptarse para detectar si pueden insertarse, eliminarse o reordenarse unidades de acceso particulares sin adaptación adicional del vídeo comprimido. Si es necesaria adaptación adicional, la unidad de aleatorización de vídeo está adaptada preferentemente para realizar esta adaptación adicional. La unidad de aleatorización de vídeo puede adaptarse también para detectar si las unidades de acceso no pueden insertarse, eliminarse o reordenarse, incluso con adaptación adicional del vídeo comprimido. Si esto se detecta, la unidad de aleatorización de vídeo puede desactivar la aleatorización para las unidades de acceso pertinentes.

La unidad de aleatorización de vídeo puede controlarse para aplicar aleatorización a primeras porciones del vídeo y no aplicar aleatorización a segundas porciones del vídeo.

La primera y segunda porciones se seleccionan preferentemente de manera aleatoria o pseudo-aleatoria.

El decodificador de vídeo puede comprender un decodificador de vídeo de hardware.

Un decodificador de hardware es típicamente un dispositivo electrónico separado (por ejemplo, un circuito integrado separado o “chip”), que está diseñado específicamente para decodificar flujos de vídeo comprimidos.

Preferentemente, el vídeo comprimido y el decodificador de vídeo cumplen con una norma de codificación de vídeo predeterminada; y la unidad de aleatorización de vídeo está adaptada para aleatorizar el vídeo comprimido de tal manera que el vídeo comprimido aleatorizado también cumple con la misma norma de codificación de vídeo.

Es decir, la aleatorización realizada mediante la unidad de aleatorización de vídeo no rompe la conformidad del flujo de bits de vídeo con la norma de codificación de vídeo. El decodificador de vídeo conforme a la norma puede entonces usarse para decodificar el flujo de bits comprimido aleatorizado.

El decodificador de vídeo y el vídeo comprimido pueden cumplir con la norma H.264 y la aleatorización puede comprender modificar el Recuento de Orden de Instantánea (POC) de una pluralidad de unidades de acceso en el vídeo comprimido H.264.

La modificación del POC preferentemente comprende al menos uno de: reordenar los valores de POC de un conjunto de unidades de acceso; y multiplicar los valores de POC de un conjunto de unidades de acceso por un número entero. Reordenar los valores de POC tiene el efecto de permutar el orden de presentación original de las unidades de acceso. Multiplicar los valores de POC por un número entero posibilita que se inserten una o más unidades de acceso adicionales entre unidades de acceso originales.

Como ya se ha mencionado anteriormente, la unidad de aleatorización de vídeo puede adaptarse para detectar si pueden insertarse, eliminarse, o reordenarse unidades de acceso particulares sin adaptación adicional del vídeo

comprimido. Si es necesaria adaptación adicional, la unidad de aleatorización está adaptada preferentemente para realizar esta adaptación adicional. En el caso de la norma H.264, esta adaptación puede comprender modificar la sintaxis del flujo de vídeo comprimido en otros aspectos (además de modificar el POC de las unidades de acceso).

La unidad de aleatorización de vídeo puede adaptarse para: detectar que el recuento de orden de instantánea está codificado implícitamente en el vídeo comprimido y, en respuesta: modificar información de encabezamiento del vídeo comprimido para señalizar que el recuento de orden de instantánea está codificado explícitamente; y asignar un valor de recuento de orden de instantánea explícito en cada unidad de acceso del vídeo comprimido.

En algunos flujos de vídeo H.264, el POC es implícito. Convertir tales flujos para codificar el POC, explícitamente hace posible aleatorizar el vídeo modificando el POC.

La unidad de aleatorización de vídeo puede adaptarse para detectar en el vídeo comprimido un modo de codificación que requeriría que el decodificador de vídeo usara el recuento de orden de instantánea para decodificar una o más unidades de acceso y, en respuesta: desactivar la aleatorización; o modificar el vídeo comprimido de modo que el decodificador de vídeo no necesite usar el recuento de orden de instantánea para decodificar la una o más unidades de acceso.

En general, el POC no es necesario en el proceso de decodificación, sino que únicamente determina el orden de presentación (visualización) de los fotogramas después de que se han decodificado. Sin embargo, en ciertos modos de codificación, la decodificación de un fotograma es dependiente de su POC. En estos modos, la aleatorización modificando el POC puede conducir a errores de decodificación - por ejemplo, cambios perceptibles en el contenido visual de los fotogramas decodificados.

Detectar un modo de este tipo puede comprender detectar un modo de predicción temporal en el que los vectores de movimiento son dependientes de valores de POC (predicción directa temporal). Como alternativa o además, detectar un modo de este tipo puede comprender detectar un modo de predicción ponderado en el que la predicción ponderada es dependiente de los valores de POC.

La unidad de aleatorización de vídeo puede adaptarse para: detectar que el vídeo comprimido está codificado de acuerdo con el perfil de línea de base de la norma H.264 y, en respuesta, modificar el vídeo comprimido para convertirlo a otro perfil H.264.

Es decir, la unidad de aleatorización de vídeo puede adaptarse para modificar la sintaxis del flujo de vídeo comprimido, para convertirlo desde un flujo de bits de perfil de línea de base conforme en un flujo de bits que cumple con un perfil diferente, preferentemente el perfil principal. En el perfil de línea de base, el orden de presentación de fotogramas es típicamente el mismo que su orden de decodificación. Por lo tanto, si un vídeo de perfil de línea de base se aleatorizó modificando el POC de ciertos fotogramas, un usuario malicioso podría desaleatorizar el vídeo presentando los fotogramas en el orden en el que se decodificaron. Convertir el vídeo a otro perfil (en particular, el perfil principal) puede ayudar a distinguir el hecho de que el orden de presentación verdadero es el mismo que el orden de decodificación.

También se proporciona un dispositivo electrónico portátil tal como un teléfono móvil o dispositivo informático de tableta, que comprende un aparato como se ha resumido anteriormente.

La presente invención puede ser particularmente beneficiosa en un dispositivo de este tipo, puesto que es común que se proporcione un reproductor de vídeo como una aplicación de software en un dispositivo de este tipo. Esta aplicación se ejecuta en un microprocesador o microcontrolador de fin general. Sin embargo, es también común que se use un acelerador de hardware (separado del procesador de fin general) para decodificar el vídeo comprimido, puesto que el procesador de fin general no tiene suficiente potencia de procesamiento. Incluso si el software de aplicación de reproductor de vídeo y el dispositivo acelerador de hardware fueran seguros (individualmente), puede ser posible para un usuario malicioso interceptar el vídeo entre la aplicación de software (procesador de fin general) y el acelerador de hardware. De acuerdo con una realización de la presente invención, el software de aplicación controla el procesador de fin general para aleatorizar el vídeo comprimido antes de enviarlo al acelerador de hardware, y controla el procesador de fin general para desaleatorizar el vídeo descomprimido aleatorizado recibido desde el acelerador de hardware. Es decir, las funciones de la unidad de aleatorización y unidad de desaleatorización pueden realizarse mediante el procesador de fin general bajo el control del software de aplicación de reproductor de vídeo.

De acuerdo con otro aspecto de la invención, se proporciona un método de decodificación de un vídeo comprimido, que comprende:

aleatorizar el vídeo comprimido, para producir un vídeo comprimido aleatorizado;
decodificar el vídeo comprimido aleatorizado, para producir un vídeo descomprimido aleatorizado; y
desaleatorizar el vídeo descomprimido aleatorizado, para producir un vídeo descomprimido desaleatorizado.

Se proporciona también un medio legible por ordenador no transitorio que almacena un programa informático, comprendiendo el programa informático medios de código de programa informático adaptados para realizar todas las etapas de un método como se ha resumido anteriormente cuando dicho programa se ejecuta en un ordenador.

La invención se describirá ahora a modo de ejemplo con referencia a los dibujos adjuntos, en los que:

la Figura 1 muestra un reproductor de video convencional ideal;
la Figura 2 muestra un reproductor de vídeo de acuerdo con una realización de la invención;
la Figura 3 muestra la unidad de aleatorización de vídeo de la Figura 2 en mayor detalle;
la Figura 4 muestra la unidad de desaleatorización de vídeo de la Figura 2 en mayor detalle;
la Figura 5 ilustra los diferentes tipos de fotograma usados de acuerdo con una realización de la invención;
la Figura 6 es un diagrama de flujo que ilustra aleatorización mediante redistribución;
la Figura 7 es un diagrama de flujo que ilustra aleatorización mediante inserción de AU;
la Figura 8 es un diagrama de flujo que ilustra aleatorización mediante sustitución de AU;
la Figura 9 es un diagrama de flujo que ilustra un procedimiento de desaleatorización;
la Figura 10 es un diagrama de flujo de pseudo-código que ilustra las etapas iniciales de un método de aleatorización para un vídeo codificado usando H.264;
la Figura 11 es un diagrama de flujo de pseudo-código que continúa desde la Figura 10, que ilustra aleatorización mediante redistribución;
la Figura 12 es un diagrama de flujo de pseudo-código que continúa desde la Figura 11, que ilustra aleatorización mediante inserción de fotograma; y
la Figura 13 es un diagrama de flujo que ilustra un método ejecutado mediante la unidad de control aleatorio de la Figura 2.

Para consumir contenido de vídeo de alta calidad HD en dispositivos inteligentes típicos, es deseable usar un decodificador de vídeo de hardware (HW) disponible en el dispositivo inteligente. Esto es debido a que un decodificador de software, que se ejecuta en un procesador de fin general en un dispositivo inteligente, puede no tener suficiente potencia de procesamiento para decodificar vídeo de HD. Un sistema típico estaría compuesto de un reproductor de vídeo (normalmente integrado en la aplicación del proveedor de servicio) de software (SW) que empleará el decodificador de vídeo de HW de la plataforma para realizar la decodificación real del flujo de vídeo comprimido.

Un ejemplo de un sistema de este tipo se ilustra en la Figura 1. Un flujo de audio y vídeo se entrega desde una fuente 100 a un reproductor de vídeo de software 110. El reproductor de vídeo de software 110 es una aplicación de software, que se ejecuta en un procesador de fin general de un dispositivo inteligente. En el reproductor de vídeo de software 110, el audio y el vídeo se manejan mediante rutas de procesamiento diferentes. El flujo de audio se desempaqueta mediante una unidad de desempaquetamiento de audio 112 y se pada a una unidad de desencriptación de audio 114 para desencriptarse. El flujo desencriptado se decodifica mediante el decodificador de audio 116 antes de pasarse al representador de audio 118. Este emite una señal de audio que se entrega a un sumidero de audio 140 (por ejemplo, un altavoz). El flujo de vídeo se desempaqueta mediante una unidad de desempaquetamiento de vídeo 122 y a continuación se desencripta mediante la unidad de desencriptación de vídeo 124. El reproductor de vídeo de software 110 a continuación envía el vídeo desencriptado a un decodificador de vídeo de hardware 130, para decodificación. El vídeo decodificado (descomprimido) se envía de vuelta desde el decodificador de vídeo de hardware 130 al reproductor de vídeo de software 110 y se representa mediante un representador de vídeo 128. La señal de vídeo emitida mediante el representador de vídeo 128 se suministra a un sumidero de vídeo 150 (por ejemplo, la pantalla de visualización del dispositivo inteligente). Como entenderán los expertos en la materia, los componentes 112-128 del reproductor de vídeo de software 110 representan unidades funcionales de la aplicación de software, todas las cuales se ejecutan en el procesador de fin general (no mostrado). Sin embargo, el decodificador de vídeo de hardware 130 es un dispositivo de hardware separado.

El reproductor de vídeo de software 110 se protege típicamente mediante mecanismos de seguridad de software, tales como: ofuscación de código, anti-depuración, anti-manipulación y anti-acceso privilegiado a la raíz. Sin embargo, hay dos puntos en esta arquitectura donde un usuario malicioso podría obtener acceso al contenido. En primer lugar, entre la unidad de desencriptación de vídeo 124 y el decodificador de vídeo de hardware 130, el vídeo está disponible desencriptado, en el dominio comprimido. En segundo lugar, entre el decodificador de vídeo de hardware 130 y el representador de vídeo 128, el vídeo está disponible desencriptado en el dominio de píxel.

Los presentes inventores han reconocido que sería deseable evitar la necesidad de un entorno confiable en la plataforma, pero aun así posibilitar la distribución segura de vídeo de alta calidad HD. La solución proporcionada de acuerdo con las realizaciones de la presente invención es poner en marcha un mecanismo de aleatorización de vídeo en el dominio comprimido y proporcionar un mecanismo de desaleatorización de vídeo en el dominio de píxel. Con este enfoque, si el contenido se obtiene mediante un usuario malicioso en los puntos débiles anteriormente identificados, será apenas visible debido a la degradación visual creada por el aleatorizador. En algunas realizaciones, para hacer el sistema más robusto a ingeniería inversa del flujo aleatorizado por un usuario malicioso, se usa un mecanismo de control aleatorio, para activar el mecanismo de aleatorización en momentos arbitrarios en el tiempo.

La Figura 2 ilustra un reproductor de vídeo de software 210 de acuerdo con una realización de la presente invención. Este está basado en la arquitectura de la Figura 1. A menos que se especifique de otra manera, los componentes compartidos son similares y no se describirán de nuevo. Las diferencias principales son que el reproductor de vídeo de software 210 comprende una unidad de aleatorización de vídeo 220, una unidad de desaleatorización de vídeo 230 y una unidad de control aleatorio 240. La unidad de aleatorización de vídeo 220 opera en el dominio comprimido. Recibe un flujo elemental comprimido de vídeo desencriptado, desde la unidad de desencriptación de vídeo 124 y genera un flujo elemental comprimido de vídeo aleatorizado como salida. Esta salida se suministra al decodificador de vídeo de hardware 130 (que puede ser idéntico al decodificador de la Figura 1). La unidad de desaleatorización de vídeo 230 opera en el dominio de píxel. Toma un vídeo descomprimido aleatorizado, recibido desde el decodificador de vídeo de hardware 130, y genera un vídeo descomprimido desaleatorizado basándose en metadatos de aleatorización. Los metadatos de aleatorización se suministran desde la unidad de aleatorización de vídeo 220 a la unidad de desaleatorización de vídeo 230, dentro del reproductor de vídeo de software 210. La salida de la unidad de desaleatorización de vídeo 230 se suministra al representador de vídeo 128 para presentación. El módulo de control aleatorio 240 controla la unidad de aleatorización de vídeo 220, para activar aleatoriamente el proceso de aleatorización.

La unidad de aleatorización de vídeo 220 y la unidad de desaleatorización de vídeo 230 están integradas en el reproductor de vídeo de software 210 y pueden protegerse mediante mecanismos de protección de software (algunos ejemplos de los cuales son conocidos en la técnica).

La unidad de aleatorización de vídeo 220 procesa las unidades de acceso (AU) en el dominio comprimido e implementa varios mecanismos de aleatorización que pueden realizarse independientemente o en combinación. Estos incluyen: reordenación de AU de vídeo (redistribución); inserción de AU de vídeo; eliminación de AU de vídeo; y sustitución de AU de vídeo. Las funciones del módulo de control aleatorio 240 y la unidad de aleatorización de vídeo 220 se ilustran en mayor detalle en la Figura 3. El módulo de control 240 comprende un generador de números aleatorios 242. Los números aleatorios que genera se usan para controlar la activación de los cuatro mecanismos de aleatorización de la unidad de aleatorización de vídeo 220. La unidad de aleatorización de vídeo 220 recibe las AU mediante una memoria intermedia de entrada 310 (que los recibe desde la unidad de desencriptación de vídeo 124. A partir de esta memoria intermedia, las AU se suministran a cuatro módulos funcionales que implementan la redistribución, inserción, eliminación y sustitución de las AU, respectivamente. Las AU aleatorizadas se emiten a continuación a una memoria intermedia de salida 330, desde donde se enviarán al decodificador de vídeo de hardware 130. Los metadatos acerca de la aleatorización se generan en los módulos funcionales. Estos metadatos de aleatorización describen las operaciones que tienen que aplicarse a las AU (es decir, cómo se han modificado en los procesos de aleatorización). Estos metadatos de aleatorización se emiten a una memoria intermedia de metadatos 340. Desde este punto, se entregan a la unidad de desaleatorización de vídeo 230, que usa los metadatos de aleatorización para reconstruir (desaleatorizar) el flujo de vídeo descomprimido correctamente. Para inserción y sustitución de las AU, necesitan proporcionarse AU adicionales. Estas se almacenan en una memoria intermedia de inserción de AU 222. En la presente realización, las AU insertadas son copias de otras AU en el flujo elemental.

El módulo de desaleatorización de vídeo 230 procesa fotogramas en el dominio de píxel e implementa varios mecanismos de desaleatorización que son dependientes de los mecanismos de aleatorización aplicados en la unidad de aleatorización de vídeo 220. Estos incluyen un redistribuidor de fotogramas; un descartador de fotogramas; y un interpolador de fotogramas. Esto se ilustra en la Figura 4. Una memoria intermedia de entrada de fotograma 410 recibe fotogramas del vídeo aleatorizado descomprimido desde el decodificador de vídeo de hardware 130. La unidad de desaleatorización de vídeo 230 toma fotogramas desde esta memoria intermedia de entrada de fotograma 410 y toma los metadatos de aleatorización desde la memoria intermedia de metadatos 340 como sus entradas. Modifica los fotogramas de manera apropiada, para invertir los efectos de la aleatorización del dominio comprimido que se aplicó anteriormente. La redistribución de fotogramas deshace la redistribución de AU realizada en la unidad de aleatorización de vídeo 220. La eliminación de fotogramas deshace la inserción de AU realizada en la unidad de aleatorización de vídeo 220. La interpolación de fotogramas deshace la eliminación de AU realizada en la unidad de aleatorización de vídeo 220. La interpolación reconstruye el fotograma perdido desde fotogramas vecinos (por ejemplo, usando técnicas conocidas para interpolación de movimiento). Como tal, el fotograma reconstruido puede no ser idéntico al fotograma codificado en la AU que se eliminó. Sin embargo, siempre que las diferencias sean pequeñas, pueden no ser fácilmente perceptibles por un observador.

Aunque puede haberse llevado a cabo cualquiera de cuatro operaciones en la unidad de aleatorización de vídeo 220, únicamente son necesarias tres unidades funcionalidades para desaleatorizar el vídeo. Esto es debido a que la sustitución de AU en la unidad de aleatorización de vídeo 220 puede verse como una combinación de eliminación de fotograma e inserción de fotograma. Por lo tanto, puede deshacerse por una combinación de eliminación de fotograma e interpolación de fotograma.

Como se usa en el presente documento, una “Unidad de Acceso” (AU) significa una subestructura lógica de un flujo elemental. El flujo elemental es un flujo de bits binario que contiene datos de vídeo codificado (es decir, comprimido). Cada fotograma es una versión decodificada (descomprimida) de una respectiva unidad de acceso.

Datos de vídeo “comprimido” significa una representación de un vídeo que no comprime datos de píxeles adecuados para presentar en una pantalla. Se prefiere una representación más compacta que los datos de píxeles en bruto (en términos del número de bits necesarios para representar el vídeo). Los datos de vídeo comprimido deben decodificarse para reconstruir datos de píxeles adecuados para representación. En algunas realizaciones el vídeo se comprime usando un método descompresión codificado por transformación de movimiento compensado basado en bloques, tal como aquellos que serán familiares para el experto en la materia.

Un ejemplo detallado se describirá ahora, que muestra cómo pueden implementarse las operaciones de aleatorización y desaleatorización, usando el códec de vídeo H.264. Los expertos en la materia apreciarán que la invención no está limitada en alcance a este códec y, entendiendo los principios desvelados a continuación, pueden construirse ejemplos similares para otros códecs.

Los métodos de aleatorización descritos a continuación son dependientes de los tipos de fotogramas en el dominio comprimido. Los expertos en la materia estarán familiarizados con los tipos habituales de fotogramas comprimidos encontrados en esquemas de codificación de movimiento compensado convencionales. Lo siguientes es un breve resumen de los tipos y sus características relevantes. Estos se ilustran en la Figura 5.

- Intra fotograma: (“fotograma I”) este tipo de fotograma se codifica sin referencia a ningún fotograma pasado o futuro. Un tipo particular de un fotograma I es un fotograma de Refresco de Decodificador Instantáneo (IDR). Ningún fotograma recibido después de un fotograma de IDR puede usar ningún fotograma antes del fotograma de IDR como un fotograma de referencia para decodificación.
- Fotograma predicho: (“fotograma P”) un fotograma que se codifica con referencia a un fotograma pasado. Este tipo de fotograma puede usarse también como un fotograma de referencia para uno o más fotogramas decodificados posteriormente.
- Fotograma predicho bidireccional: (“fotograma B”) un fotograma que se codifica con referencia a fotogramas pasados y/o futuros.

Puede extraerse una distinción entre fotogramas que se usarán como fotogramas de referencia en el proceso de decodificación y aquellos que no se usarán como fotogramas de referencia en el proceso de decodificación. En el caso de H.264, esto se aplica principalmente a fotogramas B pero también ocasionalmente a fotogramas P. Podemos hacer la distinción entre fotogramas B (y fotogramas P) que se usarán como referencias para la codificación de otros fotogramas y fotogramas B (y fotogramas P) que no se usarán como referencia para la codificación de otros fotogramas. Cuando se requiera, por motivos de claridad, los primeros se denominarán “fotogramas rB” (respectivamente “fotogramas rP”) y los últimos se denominarán “fotogramas nrB” (respectivamente “fotogramas nrP”).

Los métodos de aleatorización son de manera que el flujo resultante del mecanismo de aleatorización permanece conforme con la sintaxis del esquema de codificación. En otras palabras, permanece un flujo de bits codificado válido, de acuerdo con la definición del códec. Esto es deseable para evitar errores en el decodificador de HW y de modo que el decodificador no requiera modificación.

Los métodos de aleatorización se aplican en el dominio comprimido pero son reversibles en el dominio de píxel. Realizar la aleatorización no requiere decodificación de los fotogramas recibidos - en su lugar, la aleatorización opera en elementos de sintaxis de nivel superior del flujo de vídeo comprimido.

El recuento de orden de instantánea (POC) se usa en el proceso de decodificación para identificar el orden de presentación (visualización) de cada uno de los fotogramas decodificados. Puesto que el uso de la predicción bidireccional del orden de fotograma decodificado es diferente del orden de visualización - los fotogramas de referencia para un fotograma B deben decodificarse antes del mismo fotograma B, incluso aunque el fotograma B tenga lugar en un instante de presentación anterior. El POC normalmente está embebido en la sintaxis de alto nivel de un fotograma codificado y se usa en la última etapa del proceso de decodificación, para reordenar apropiadamente los fotogramas de acuerdo con su orden de visualización. Los procesos de aleatorización descritos a continuación manipulan el POC de una pluralidad de fotogramas (opcionalmente además de otras modificaciones).

La aleatorización mediante reordenación se ilustra en la Figura 6. El principio es reorganizar el orden de presentación de algunos de los fotogramas, cambiando el valor de POC embebido en cada unidad de acceso. Si no se compensa (en la unidad de desaleatorización de vídeo 230) esto crea molestia visual en forma de fuerte vibración. Cuanto más movimiento esté presente en el contenido de vídeo, más pronunciada será la molestia. El proceso de desaleatorización correspondiente reordena los fotogramas apropiadamente. Una restricción se aplica para las AU que codifican fotogramas de referencia: si se usa una AU re-ordenada como referencia, su índice de referencia deberá mantenerse de modo que las AU que lo usan como una AU de referencia en su proceso de decodificación se decodifican apropiadamente.

El procedimiento de redistribución empieza en la etapa 610, en la que la unidad de aleatorización de vídeo 220 comprueba si se ha activado la redistribución mediante la unidad de control aleatorio 240. Si es así, la siguiente AU se lee desde la memoria intermedia de entrada 310, en la etapa 620. A continuación, en la etapa 630, la unidad de

aleatorización de vídeo 220 comprueba si el POC puede cambiarse sin modificación adicional de la AU. Si es así, el método continúa a la etapa 660 y el valor de POC se cambia. Si el POC no puede cambiarse sin adaptación adicional de la AU, el método continúa a la etapa 640 y la unidad de aleatorización comprueba si el POC puede cambiarse en conjunto con la adaptación adicional de la AU. Si es así, el método continúa a la etapa 650 y se realiza la adaptación adicional necesaria de la AU. El método a continuación continúa a la etapa 660. Después de la etapa 660, la AU con su valor de POC modificado se escribe en la memoria intermedia de salida 330, en la etapa 670. En la etapa 610, si la unidad de aleatorización de vídeo 220 determina que no se ha activado la redistribución, el método continúa a la etapa 690. El método puede continuar también a la etapa 690 si se determina en la etapa 640 que el POC no puede cambiarse (incluso con adaptación adicional de la AU). En la etapa 690, la unidad de aleatorización de vídeo 220 comprueba si el POC debe adaptarse debido a que se ha realizado aleatorización en las AU anteriores. (La aleatorización aplicada a los valores de POC de las AU anteriores puede tener un efecto en cadena en los valores de POC de AU posteriores, incluso si estas AU posteriores no se aleatorizan ellas mismas). En este caso, el método continúa a la etapa 660, para actualizar los valores de POC. De otra manera, el método continúa directamente a la etapa 670, para escribir la AU en la memoria intermedia de salida 330. Finalmente, en la etapa 680, se escriben metadatos de aleatorización a la memoria intermedia de metadatos 340. El procedimiento se repite para cada unidad de acceso. Después de la unidad de acceso final, el procedimiento de redistribución finaliza.

La aleatorización mediante inserción de AU se ilustra en la Figura 7. El principio es insertar AU entre AU existentes en el dominio comprimido. Si el contenido de las AU insertadas está relativamente descorrelacionado con las AU adyacentes creará discontinuidades visuales fuertes, provocando molestias (si no se eliminan mediante la desaleatorización apropiada). Cuanto menos movimiento esté presente en el contenido mayor molestia se creará. El proceso de desaleatorización comprende eliminar los fotogramas adicionales insertados. Preferentemente, las AU insertadas no deberían ser fácilmente identificables (para un usuario malicioso). De manera ideal, la AU insertada debería ser diferente en cada inserción. Una restricción es que las AU insertadas no deberán usarse como una referencia para la decodificación de los fotogramas codificados originales en el flujo. Si un fotograma original tuviera que decodificarse usando un fotograma insertado como un fotograma de referencia, su proceso de decodificación se corrompería. Para evitar esto en la presente realización, el sistema asegura que los fotogramas insertados no entren en la memoria intermedia de referencia, usando únicamente una AU de nrB o una de nrP presente en el flujo. Como alternativa o además, el sistema podría insertar un conjunto continuo de AU cuya decodificación es autónoma. Sin embargo, en el último caso, esto puede realizarse únicamente inmediatamente antes de un fotograma de IDR, como se ejemplifica a continuación.

La estructura de flujo de entrada es:

IDR P P P P P P B P B P IDR P B P B

La estructura de flujo de salida, después de la inserción de un Grupo de Instantáneas (GOP) autónomas [IDR P P P] antes del segundo fotograma de IDR, es:

IDR P P P P P P B P B P [IDR P P P] IDR P B P B

Puesto que el GOP se inserta inmediatamente antes de un fotograma de IDR, ninguno de los fotogramas posteriores hará referencia a fotogramas en el GOP insertados como fotogramas de referencia. Para que esa decodificación del GOP insertado sea autónoma, el mismo GOP insertado comienza también con un fotograma de IDR.

Con referencia a la Figura 7, el proceso se inserción comienza y se lee una AU desde la memoria intermedia de entrada 310, en la etapa 710. A continuación, en la etapa 715, la unidad de aleatorización de vídeo 220 comprueba si se ha activado inserción de AU mediante la unidad de control aleatorio 240. Si se ha activado la inserción de AU, el método continúa a la etapa 720 y la unidad de aleatorización de vídeo 220 comprueba si puede insertarse una AU sin adaptación adicional del flujo elemental. Si es así, el método continúa a la etapa 735 y la AU a insertarse se lee desde la memoria intermedia de inserción de AU 222. El POC de la AU insertada se actualiza en la etapa 740. Después de eso, el método continúa a la etapa 760 y la unidad de aleatorización comprueba si el POC debe actualizarse debido a la aleatorización que se realizó en AU anteriores. Si es así, el POC se actualiza en la etapa 765 y el método continúa a la etapa 745. Si no, el método continúa directamente a la etapa 745. En la etapa 745, si se ha insertado una AU, la AU insertada se escribe en la memoria intermedia de salida 330. El método a continuación continúa a la etapa 750, en la que la AU original (leída desde la memoria intermedia de entrada en la etapa 710) se escribe en la memoria intermedia de salida 330. Finalmente, los metadatos de aleatorización se escriben en la memoria intermedia de metadatos 340, en la etapa 755. En la etapa 715, si la unidad de aleatorización determina que no se ha activado esa inserción de AU, el método continúa a la etapa 760. En la etapa 720, si la unidad de aleatorización 220 determina que una AU no puede insertarse sin adaptación adicional del flujo elemental, el método continúa a la etapa 725. En la etapa 725, la unidad de aleatorización de vídeo 220 comprueba si puede insertarse una AU en conjunto con adaptación adicional del flujo de bits. Si es así, la adaptación necesaria se realiza en la etapa 730 y el método continúa a la etapa 735. Si una AU no puede insertarse incluso con adaptación adicional del flujo, el método continúa desde la etapa 725 a la etapa 760. Este procedimiento se realiza para cada AU en la secuencia. Cuando se han procesado todas las AU, el procedimiento finaliza.

La aleatorización puede comprender también eliminación de las AU, que es la inversa de insertar las AU y es relativamente fácil de implementar. Las AU se leen desde la memoria intermedia de entrada 310 y se copian a la memoria intermedia de salida 330 a menos que se tengan que eliminar. Es decir, únicamente las AU que no se eliminan se escriben en la memoria intermedia de salida 330. En general, no hay necesidad de cambiar el valor de POC de cada AU, excepto cuando necesita actualizarse como consecuencia de aleatorización que se aplicó a AU anteriores en el vídeo. Se crea molestia visual puesto que se reduce la velocidad de fotogramas del contenido. El proceso de desaleatorización comprende reconstruir los fotogramas que corresponden a las AU perdidas, en el dominio de píxel, por medio de interpolación de fotograma basada en movimiento. Puesto que la reconstrucción por interpolación no es perfecta, la representación final puede contener algunos artefactos visuales relacionados con el proceso de re-interpolación. Preferentemente, las AU que se usarán como fotogramas de referencia para decodificar otros fotogramas no se eliminan. Es decir, únicamente se eliminan AU de nrB o nrP.

La aleatorización mediante sustitución de AU se ilustra en la Figura 8. El principio es combinar eliminación de AU e inserción de AU. Con referencia a la Figura 8, cuando se inicia el proceso de aleatorización de sustitución de AU, se lee una AU desde la memoria intermedia de entrada 310, en la etapa 805. A continuación, la unidad de aleatorización de vídeo 220 comprueba, en la etapa 810, si la sustitución de AU se ha activado por la unidad de control aleatorio 240. Si es así, el método continúa a la etapa 820. En la etapa 820, la unidad de aleatorización de vídeo 220 comprueba si esta AU puede sustituirse sin adaptación adicional del flujo. Si es así, el método continúa a la etapa 835 y se lee la AU a insertarse en lugar de la AU actual desde la memoria intermedia de inserción de AU 222. El método a continuación continúa a la etapa 840. En este punto, el POC de la AU de fotograma insertada se actualiza. A continuación, en la etapa 845, la AU (insertada) se escribe en la memoria intermedia de salida 330. Finalmente, en la etapa 850, los metadatos de aleatorización correspondientes se escriben en la memoria intermedia de metadatos 340. Si la unidad de aleatorización determina, en la etapa 820, que la AU no puede sustituirse sin adaptación adicional del flujo, el método continúa a la etapa 825 y la unidad de aleatorización comprueba si la AU puede sustituirse en conjunto con adaptación adicional. Si es así, el método continúa a la etapa 830 y se realiza la adaptación adicional necesaria. Después de eso, el método continúa a la etapa 835, como anteriormente. En la etapa 810, si no se ha activado sustitución de AU, el método continúa a la etapa 855 y la unidad de aleatorización de vídeo 220 comprueba si el POC necesita actualizarse debido a que se realizó aleatorización en AU anteriores. Si es así, el método continúa a la etapa 840 y a continuación a la etapa 845. Si no, el método continúa directamente a la etapa 845, en la que la AU (original) se escribe en la memoria intermedia de salida. En la etapa 825, si la AU actual no puede sustituirse incluso en conjunto con adaptación adicional del flujo, el método también continúa a la etapa 855. Después de que se ha realizado el proceso para cada AU, el proceso finaliza.

La Figura 9 resume el proceso de desaleatorización. Cuando la desaleatorización comienza, la unidad de desaleatorización de vídeo 230 lee metadatos de aleatorización desde la memoria intermedia de metadatos 340, en la etapa 910. A continuación, en la etapa 915, la unidad de desaleatorización de vídeo 230 comprueba (usando los metadatos) si se realizó redistribución de POC mediante la unidad de aleatorización de vídeo 220. Si ha tenido lugar redistribución de POC, el método continúa a la etapa 920 y la unidad de desaleatorización de vídeo 230 lee el fotograma correcto desde la memoria intermedia de entrada de fotograma 410. Desde este punto, el método continúa a la etapa 960 y el fotograma se escribe en una memoria intermedia de salida de fotograma 420. Si no hubo redistribución de POC, el método continúa a la etapa 925 y la unidad de desaleatorización comprueba si se ha insertado una AU. Si es así, el método continúa a la etapa 930 y se descarta el siguiente fotograma en la memoria intermedia de entrada de fotograma 410. Si no se ha insertado AU, el método continúa en su lugar a la etapa 935 y la unidad de desaleatorización de vídeo 230 comprueba si se ha eliminado una AU. Si es así, el fotograma faltante se reconstruye mediante interpolación, en la etapa 940. Desde este punto, el método continúa a la etapa 960 y el fotograma interpolado se escribe en la memoria intermedia de salida de fotograma 420. Si no se ha eliminado una AU, el método continúa en su lugar a la etapa 945 y la unidad de desaleatorización de vídeo 230 comprueba si se ha sustituido una AU. Si es así, el método continúa a la etapa 950. El siguiente fotograma en la memoria intermedia de entrada de fotograma 410 se descarta y el fotograma faltante se reconstruye mediante interpolación. Una vez más, el método continúa a la etapa 960 y el fotograma interpolado se escribe en la memoria intermedia de salida de fotograma 420. Si no ha habido sustitución de AU, el método continúa desde la etapa 945 a la etapa 955. Esto significa que no se realizó aleatorización en esta parte del vídeo. El siguiente fotograma se lee desde la memoria intermedia de entrada de fotograma 410 y el método continúa directamente a la etapa 960, en la que este fotograma se escribe en la memoria intermedia de salida de fotograma 420. Después de que, o bien, todas los fotogramas se han descartado en la etapa 930, o bien, se han escrito en la memoria intermedia de salida en la etapa 960, el proceso finaliza.

Una implementación específica de los procesos de aleatorización se describirá ahora en mayor detalle, para un vídeo comprimido codificado de acuerdo con la norma H.264. En particular, se proporcionarán ejemplos de cómo implementar aleatorización redistribuyendo y aleatorizando mediante inserción de fotograma, para un flujo codificado H.264. Cuando se usa una variable sin definirse de otra manera en este punto, puede suponerse que la variable se define en la norma H.264 y que está presente en el flujo de bits de vídeo comprimido. La norma H.264 se define en la Recomendación de la ITU-T de H.264, “Advanced video coding for generic audiovisual services”. Esta es la misma que la norma conocida como MPEG-4 AVC, definida en ISO/IEC 14496-10, “Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding”. En caso de cualquier ambigüedad, deberían tomarse las referencias a la norma para hacer referencia a la versión, revisión o edición de la norma vigente en la fecha de

prioridad de la presente invención. Sin embargo, es habitual que las últimas versiones de tales normas sean compatibles hacia atrás con versiones anteriores - las últimas versiones amplían la norma, mientras mantienen los elementos normativos de las versiones anteriores. Por lo tanto, se ha de esperar que no surja ambigüedad y que un flujo de bits que cumpla con cualquier versión futura de la norma también cumpla con la versión vigente en la fecha de prioridad.

Alguna información preliminar es útil para entender los detalles de cómo se implementa la aleatorización.

El Recuento de Orden de Instantánea (POC) se usa en el proceso de decodificación de H.264 para identificar el orden de visualización (es decir, orden de presentación) de cada fotograma decodificado. Para predicción bidireccional, en particular, el orden en el que se decodifican los fotogramas puede ser diferente del orden en el que se visualizan. Una manera de implicar una redistribución de los fotogramas decodificados es modificar el POC en el flujo codificado. El POC de cada AU se codifica explícitamente en el flujo (modo 0) o se determina implícitamente durante el proceso de decodificación (modo 1 y modo 2). Puesto que la aleatorización cambiará el POC en el flujo, si la decodificación es implícita es necesario convertir desde uno de los modos implícitos (modo 1 y modo 2) al modo explícito (modo 0). Para detalles adicionales de cómo se recupera el POC durante el proceso de decodificación H.264, se hace referencia al lector al documento “H.264/AVC Frame and picture management” (Iain G. Richardson, enero de 2004), disponible en línea en: www4.rgu.ac.uk/files/avc_picmanagement_draft1.pdf. Esta información ya será familiar para los expertos en la materia.

Hay dos modos de decodificación en los que la etapa de compensación de movimiento realizada por el decodificador de vídeo de hardware 130 será dependiente del valor de POC. Estos son los modos de “predicción directa temporal” y “predicción ponderada”. Puesto que el POC no se usa simplemente para determinar el orden de presentación, en estos modos, su uso debe detectarse por la unidad de aleatorización de vídeo 220, para tenerlos en cuenta para la modificación del POC.

El perfil de línea de base H.264 es un perfil específico donde no hay predicción bidireccional. En ese caso, el flujo de vídeo codificado consiste exclusivamente en fotogramas I y fotogramas P. El orden de decodificación normalmente será idéntico al orden de visualización. En consecuencia, para un flujo de perfil de línea de base, normalmente será más fácil para un usuario malicioso detectar y deshacer cualquier redistribución del orden de visualización de los fotogramas decodificados. Una manera para superar el problema es convertir el flujo de perfil de línea de base en un flujo de perfil principal. Esto puede hacerse simplemente cambiando la semántica de alto nivel del flujo - por ejemplo, modificando información de encabezamiento. No es necesario decodificar y volver a codificar el vídeo para convertir desde línea de base a perfil principal.

La implementación del proceso de aleatorización se describirá ahora. Por claridad y simplicidad, la explicación a continuación no menciona la activación y desactivación de la aleatorización mediante el módulo de control aleatorio.

Si la señalización del POC es implícita (modo 1 o modo 2) el flujo de bits de vídeo comprimido se modifica de modo que la señalización se haga explícita (modo 0). Esto puede realizarse en dos etapas. En primer lugar, la estructura de datos sequence_parameter_set se modifica en dos aspectos:

pic_order_cnt_type = 0
log2_max_pic_order_cnt_lsb_minus4 = 4

Sequence_parameter_set (SPS) es una estructura de datos de encabezamiento definida en la norma H.264. Por conveniencia del lector se reproduce en el apéndice a continuación. Los parámetros en este encabezamiento definen cómo se codifican las AU posteriores en el flujo (hasta la siguiente aparición de sequence_parameter_set). Obsérvese que el valor de log2_max_pic_order_cnt_lsb_minus4 deberá estar en el intervalo de 0 a 12, inclusivo.

En segundo lugar, para cada AU, modificar la estructura de datos slice_header para asignar un valor de POC explícito en pic_order_cnt_lsb. El slice_header es una estructura de datos de encabezamiento definida en H.264, que describe cómo se codifica un “corte”. Típicamente, hay un corte para cada AU, pero es posible codificar una AU como una pluralidad de cortes, en los que cada corte representa una porción del fotograma de vídeo.

Para realizar la aleatorización, el flujo de vídeo comprimido se analiza AU a AU. El método de aleatorización opera en grupos de AU consecutivas del mismo tipo. Cada grupo de este tipo se indicará en el presente documento como un “edit_gop”. Los valores de POC de las AU en el edit_gop se manipulan y se insertan fotogramas en el edit_gop. Limitar la manipulación de POC a un grupo de AU consecutivas del mismo tipo ayuda a evitar una situación en la que las AU hacen referencia a las AU de referencia incorrecta, en el proceso de decodificación. En la presente realización, cada edit_gop está limitado en tamaño, para contener un máximo de 8 AU consecutivas. Esto es para limitar el retardo de procesamiento que podría introducirse por la aleatorización y desaleatorización. Si tienen lugar más de 8 AU del mismo tipo consecutivamente en el flujo comprimido, entonces estas se subdividirán en varios edit_gop consecutivos del mismo tipo.

Lo siguiente es un ejemplo. Considérese la siguiente estructura de flujo:

I nrB nrB P nrB nrB P P P B B

Este flujo se dividirá en 6 edit_gop:

Edit_gop_1: I
Edit_gop_2: nrB nrB
Edit_gop_3: P
Edit_gop_4: nrB nrB
Edit_gop_5: P P P
Edit_gop_6: B B

Las siguientes variables se usan en el transcurso del proceso de aleatorización:

- Edit_gop_type: indica el tipo de AU (por ejemplo, I, rP, nrP, rB, nrB) de las AU consecutivas en el edit_gop.
- Edit_gop_size: el número de AU consecutivas del mismo tipo que constituyen el edit_gop.
- Edit_gop_count: el número de edit_gop procesados hasta ahora. Esto se usa en el caso de inserción de fotograma, para refrescar el origen del fotograma nrB que se está insertando en el flujo. En otras palabras, después de que se ha procesado un número especificado de edit_gop, una nueva AU nrB desde el flujo de vídeo comprimido se copiará en la memoria intermedia de inserción de AU 222.
- Get_nrb_to_insert: variable booleana (es decir, una variable binaria, que puede asumir el valor “VERDADERO” o el valor “FALSO”). Cuando es VERDADERO, esto indica que debería copiarse un fotograma nrB a la memoria intermedia de inserción de AU 222 para uso posterior como una AU insertada.
- First_AU_in_edit_gop: variable booleana que, cuando es VERDADERO, indica que la AU actualmente procesada es la primera de edit_gop.
- nextAU_type: indica el tipo de la AU en el flujo que sigue a la AU que se está procesando actualmente.
- Direct_spatial_mv_pred_flag: un parámetro H.264 presente en el flujo, que define el tipo de predicción directa usada en predicción de movimiento. Para explicación adicional, véase también: http://wiki.multimedia.cx/index.php?title=Motion_Prediction
- Nr_gop: se refiere a un edit_gop constituido por AU no de referencia
- Sub_edit_gop: un subconjunto de un edit_gop excluyendo la primera AU y la última AU del edit_gop (en algunos casos es preferible no aleatorizar las AU al inicio y final de un edit_gop, respectivamente).
- POC_list: una lista de todos los valores de POC de las AU que constituyen el sub_edit_gop.
- New_poc_list: esta contiene una redistribución de la lista de POC.
- Poc_org: el valor de POC original de una AU, referenciado en poc_list
- Poc_new: el valor de POC redistribuido de una AU, referenciado en New_poc_list
- Org_flag: valor booleano que indica si la AU actual estaba en el flujo original. Este se usa para distinguir AU insertadas desde AU originales, en el flujo aleatorizado de salida.

La Figura 10 es un diagrama de flujo de pseudo-código que ilustra las primeras etapas del proceso de aleatorización realizadas mediante la unidad de aleatorización de vídeo 220, de acuerdo con la presente realización. En primer lugar, en la etapa 1010, las variables necesarias para realizar aleatorización se inicializan. A continuación, en la etapa 1020, se construye el edit_gop, usando las AU recibidas secuencialmente mediante la memoria intermedia de entrada 310. Para preparar la inserción de fotograma, el valor de POC de cada AU se multiplica por 2. La primera AU de nrB detectada se graba en la memoria intermedia de inserción de AU 222, para uso más tarde como un fotograma insertado. La construcción del edit_gop continúa hasta que se alcance el tamaño máximo (8 AU), o se recupere una AU de un tipo diferente de la memoria intermedia de entrada 310.

La Figura 11 es un diagrama de flujo de pseudo-código que ilustra la aleatorización del edit_gop mediante redistribución. Si edit_gop contiene solamente una AU, no puede redistribuirse. Análogamente, si todas las AU en el edit_gop usan predicción directa temporal (Direct_spatial_mv_pred_flag==1), no se realiza redistribución. Si edit_gop consiste en fotogramas nrB o nrP (es decir, si edit_gop es un nr_gop) todas las AU en el edit_gop pueden redistribuirse. De otra manera, el primer fotograma y el último fotograma del edit_gop se excluyen de la redistribución. En este último caso, el edit_gop debe contener más de 3 fotogramas para que la redistribución sea posible. La redistribución se consigue haciendo a new_poc_list una permutación aleatoria de los valores de POC en poc_list. En otras palabras, el conjunto original de los valores de POC se vuelve a asignar a las diversas AU en un orden aleatorio. Los valores de poc_org y poc_new para cada AU en el edit_gop se escriben en la memoria intermedia de metadatos 340. Org_flag se establece a VERDADERO y esta variable se escribe también en la memoria intermedia de metadatos 340, para cada AU.

La Figura 12 es un diagrama de flujo de pseudo-código que ilustra la aleatorización de edit_gop mediante inserción de fotograma. La inserción de fotograma se realiza en la etapa 1210, y la unidad de aleatorización prepara para el siguiente edit_gop en la etapa 1220.

En la etapa 1210, la unidad de aleatorización de vídeo 220 selecciona aleatoriamente posiciones en la secuencia de AU donde debieran insertarse los fotogramas adicionales. En cada posición seleccionada, la AU de nrB que se almacena en la memoria intermedia de inserción de AU 222 se inserta en la secuencia. Como alternativa, la AU a insertarse puede generarse automáticamente - es decir, puede insertarse una AU sintética en lugar de insertar una copia de otra AU desde el mismo flujo. El valor de POC de la AU insertada se actualiza con el valor de POC correcto. Los valores de poc_org y poc_new para cada AU insertada se escriben en la memoria intermedia de metadatos 340. Org_flag se establece a FALSO y esta variable se escribe también en la memoria intermedia de metadatos 340, para cada AU insertada.

En la etapa 1220, edit_gop se escribe en la memoria intermedia de salida 330. La unidad de aleatorización de vídeo 220 comprueba si hay alguna AU más para procesar. Si no, el proceso termina. Si hay aún más AU para procesar, las variables usadas en la aleatorización se reinicializan. Si se han procesado 20 edit_gop desde que se refrescaran por última vez los contenidos de la memoria intermedia de inserción de AU, la bandera get_nrB_to_insert se establece a VERDADERO. El proceso a continuación vuelve a la etapa 1020 en la Figura 10.

En ciertos casos, debería tenerse cuidado especial, para evitar interferir con la decodificación del vídeo comprimido aleatorizado.

En un primer ejemplo, se debe tener cuidado especial cuando se maneja un flujo que incorpora predicción temporal. El parámetro H.264 direct_spatial_mv_pred_flag especifica el método usado en el proceso de decodificación para derivar vectores de movimiento (MV) e índices de referencia para inter predicción como sigue:

- Si direct_spatial_mv_pred_flag es igual a 1, el proceso de derivación para vectores de movimiento de luma para dB_Skip, B_Direct_16x16 y B_Direct_8x8 deberá usar predicción de modo directo espacial. En este caso, la predicción directa es espacial y no depende de valores de POC de AU. Esto significa que la redistribución de POC es posible y la inserción de fotogramas es posible.
- De otra manera - es decir, si direct_spatial_mv_pred_flag es igual a 0 - los valores de POC del primer fotograma de referencia en ambas listas de referencia (Lista0 y Lista1) y el valor de POC de los fotogramas B están implicados en el proceso de calcular el valor de MV cuando el tipo de macro-bloque (MB) es uno de: B_Skip, B_Direct_16x16 o B_Direct_8x8. (Las listas de referencia son listas de fotogramas de referencia para el proceso de decodificación, como se especifica mediante la norma H.264). En este caso, la predicción directa es temporal y los valores de POC de AU están implicados en el cálculo de vector de movimiento. Esto significa que la redistribución de POC es posible en ciertos casos, y la inserción de fotograma es posible bajo la condición de que todos los valores de POC se multiplicaran por 2 originalmente.

En particular, para redistribución de POC en el caso de predicción directa temporal (es decir, cuando direct_spatial_mv_pred_flag es igual a 0):

- Para un nr_gop que consiste en fotogramas nrB, los valores de POC no deberían redistribuirse;
- Para un r_gop que consiste en fotogramas B, los valores de POC no deberían redistribuirse;
- Para un r_gop que consiste en fotogramas I/IDR/P, el algoritmo de modificación de POC podría aplicarse aún - en principio - a algunas AU del edit_gop, puesto que únicamente la primera y la última AU en el edit_gop pueden implicarse en el proceso de calcular valores de MV, dentro del decodificador. Sin embargo, el valor de direct_spatial_mv_pred_flag está presente en el slice_header, por lo que puede cambiar para cada corte en el edit_gop. En principio, sería posible redistribuir los valores de POC en un edit_gop de este tipo, pero esto requeriría comprobar el valor de direct_spatial_mv_pred_flag para cada corte y calcular apropiadamente la redistribución del POC. Por lo tanto, para evitar introducir retardo excesivo en el sistema, el presente método no redistribuye valores de POC si alguno de los recortes en el edit_gop tienen un direct_spatial_mv_pred_flag que es igual a 1.

Para inserción de fotograma, se aplica un único procedimiento independientemente del valor de direct_spatial_mv_pred_flag. El valor de POC de cada fotograma se multiplica por 2, que posibilita que la unidad de aleatorización 220 inserte fotogramas sin afectar el cálculo de vector de movimiento en el decodificador, para cualquier valor de direct_spatial_mv_pred_flag y para cualquier tipo de edit_gop.

En un segundo ejemplo necesita tenerse cuidado especial cuando se maneja un flujo que incorpora predicción ponderada. Hay tres posibles modos de predicción ponderada definidos en la norma H.264. El valor de weighted_bipred_idc deberá ser 0, 1 o 2. Estos modos se definen como sigue:

- weighted_bipred_idc igual a 0 especifica que la predicción ponderada por defecto deberá aplicarse a cortes B.
- weighted_bipred_idc igual a 1 especifica que la predicción ponderada explícita deberá aplicarse a cortes B.
- weighted_bipred_idc igual a 2 especifica que la predicción ponderada implícita deberá aplicarse a cortes B.

En los primeros dos casos (0 y 1), la predicción no es dependiente de los valores de POC de otros fotogramas y puede realizarse redistribución de POC e inserción de fotograma.

Para redistribución de POC en el tercer caso, cuando weighted_bipred_idc es igual a 2, cambiar el valor de POC sin adaptación adicional provocaría que las instantáneas se decodificaran incorrectamente. Para evitar esto, se realiza adaptación adicional del flujo de bits. En particular, la información de encabezamiento del flujo comprimido se modifica, convirtiendo el modo de predicción ponderado desde un modo implícito a un modo explícito. Esto se hace en las siguientes etapas:

- Modificar el Conjunto de Parámetros de Instantánea (PPS), estableciendo weighted_bipred_flag a 1,
- Calcular una weighted_table para cada fotograma e insertarla en el flujo,
- Modificar el slice_header, e indicar a qué weighted_table calculada deberá hacer referencia para predicción ponderada.

Para inserción de fotograma, se aplica un único procedimiento independientemente del valor de weighted_bipred_idc. El valor de POC de cada fotograma se multiplica por 2, que posibilita que la unidad de aleatorización de vídeo 220 inserte fotogramas sin afectar la decodificación, para cualquier valor de weighted_bipred_idc y para cualquier tipo de edit_gop.

Preferentemente, debería tenerse cuidado especial cuando se manejan flujos de perfil de línea de base. Para aleatorizar un flujo de perfil de línea de base, la unidad de aleatorización modifica la sintaxis de alto nivel del flujo de modo que se haga un flujo de perfil principal válido. Esta modificación comprende las siguientes etapas:

- Modificar el SPS, para distinguir el flujo de perfil de línea de base como un flujo de perfil principal; y
- Cambiar algunos fotogramas P en fotogramas B.

En el perfil de línea base, no hay fotogramas B y generalmente el POC evolucionará de una manera incremental. Este no es el caso en el perfil principal donde el POC no es incremental entre fotogramas, debido a las diferentes ordenaciones usadas para decodificación de las AU y presentación de los fotogramas. En consecuencia, en el caso de un flujo de perfil de línea de base, si la unidad de aleatorización simplemente modifica el POC, puede ser fácil para un usuario malicioso detectar esto y deshacer la aleatorización. Los fotogramas P en el flujo de perfil de línea de base pueden convertirse para parecer como fotogramas B en el flujo de perfil principal sin recodificación y con únicamente una modificación de sintaxis de alto nivel. Una vez que se ha adaptado la sintaxis del flujo de esta manera, el proceso de aleatorización puede aplicarse como ya se ha descrito anteriormente. Los valores de POC de las AU en el flujo pueden modificarse con una probabilidad reducida de que un usuario malicioso podrá identificar e invertir el proceso de aleatorización, simplemente consiguiendo acceso a la salida del decodificador de vídeo de hardware 130.

Para posibilitar la desaleatorización correcta del flujo decodificado en la unidad de desaleatorización de vídeo 230 antes de representación, se genera un conjunto de metadatos durante el proceso de aleatorización (redistribución de POC o inserción de fotograma, en los ejemplos anteriormente detallados). Para cada AU que va en la memoria intermedia de salida 330, la unidad de aleatorización de vídeo 220 proporciona los siguientes metadatos asociados:

- poc_org: un número entero que se establece al valor de POC original (para una AU existente) o se establece a -1 (para un fotograma insertado)
- poc_new: un número entero que es el valor de POC modificado después de la aplicación del proceso de aleatorización
- org_flag; un valor booleano (representado por un número entero). Cuando este se establece a 0 indica un fotograma insertado, que debería descartarse para representación. Cuando se establece a 1, indica un fotograma original, que debería representarse.

Los procesos de desaleatorización no se describirán en detalle, puesto que simplemente invierten el proceso de aleatorización, que se ha descrito en detalle anteriormente. Los expertos en la materia, entenderán fácilmente mediante esta etapa, cómo implementar los procesos de desaleatorización correspondientes.

La descripción anterior se ha concentrado en las modificaciones de nivel de flujo que se usan para implementar la aleatorización. En algunos casos, pueden requerirse modificaciones de nivel de sistema también. Por ejemplo, en algunos decodificadores de vídeo conocidos, la entrada al decodificador comprende una Indicación de Tiempo de Composición (CTS) además de la AU. La indicación de tiempo de composición es una referencia de tiempo que indica el instante en el que un fotograma de decodificación deberá presentarse en la pantalla. (En sistemas MPEG-2, esto se conoce también como la indicación de tiempo de presentación: PTS.) La CTS normalmente se proporciona en la capa de sistema que encapsula el flujo. En un sistema ideal, la CTS y el POC están correlacionados en tiempo. Si el decodificador usa la CTS, además de las AU, para decodificar el vídeo, entonces debería modificarse la CTS en el proceso de aleatorización, de modo que sea coherente con el valor de POC de cada AU. Esto evitará errores en el decodificador, debido a la inconsistencia entre valores de CTS y POC.

En este caso, los metadatos de aleatorización para cada AU se ampliarán con las dos siguientes variables (cada una de tipo int largo) para asegurar desaleatorización apropiada:

- cts_org: el valor de CTS original o -1 para un fotograma insertado
- cts_new: el valor de CTS modificada después de la aplicación del proceso de aleatorización

Como se ha mencionado anteriormente, la descripción de los procesos de aleatorización anteriores no tuvo en cuenta el uso de la unidad de control aleatorio 240, que activa y desactiva la aleatorización a tiempos aleatorios. Sin embargo, los detalles esenciales de la aleatorización son los mismos cuando se usa el módulo de control aleatorio 240. Cuando la aleatorización está desactivada, para una AU dada, el valor poc_new será igual al valor poc_org y org_flag será igual a 1.

Un método ejemplar ejecutado mediante la unidad de control aleatorio 240 se describirá ahora, con referencia al diagrama de flujo de la Figura 13. La unidad de control aleatorio puede operarse para activar y desactivar aleatoriamente la aleatorización. La Figura 13 ilustra el proceso de activación/desactivación para un método de aleatorización. Este proceso se replica para cada método de aleatorización. El método se inicia y la unidad de control aleatorio 240 selecciona aleatoriamente un número N de 8 bits, entre 0 y 255 (excluyendo el 0 y 255). Este número de 8 bits se genera mediante el generador de números aleatorios 242 de la unidad de control aleatorio 240. A continuación, en la etapa 1315, la unidad inicia un temporizador T de granularidad To. En la etapa 1320, la unidad de control aleatorio lee el bit menos significativo b del número N. En la etapa 1325, comprueba si b es igual a 1. Si es así, se activa la aleatorización, en la etapa 1330. Si no (es decir, b es igual a 0), se desactiva la aleatorización, en la etapa 1335. En cualquier caso, el método a continuación continúa a la etapa 1340 y espera T_0 segundos antes de continuar a la etapa 1345. En esta etapa, se añade T_0 a la variable del temporizador T. La unidad de control aleatorio a continuación comprueba, en la etapa 1350, si T es mayor que 255 T_0 veces. Si lo es, el método se mueve a la etapa 1310 y se selecciona aleatoriamente un nuevo número N. Si no, el método se mueve a la etapa 1355 y el número existente N se desplaza a la derecha en un bit. Desde la etapa 1355, el método vuelve a la etapa 1320.

T_0 se elige para que esté en el orden de segundos, de modo que la aleatorización puede aplicarse de una manera significativa antes de que se desactive de nuevo. Opcionalmente, To puede también aleatorizarse, para hacer que sea más difícil para un usuario malicioso detectar la activación y desactivación de la aleatorización. Para el mismo fin, el número N puede elegirse opcionalmente a partir del subconjunto de números enteros entre 0 y 255 que contienen no más de dos bits idénticos adyacentes (en forma binaria de N). Por ejemplo, se permitiría 0b00101101, pero no 0b11100100. Esto ayuda a evitar que los medios de aleatorización permanezcan activados o desactivados durante un periodo continuo largo.

Aunque la invención se ha ilustrado y descrito en detalle en los dibujos y descripción anterior, tal ilustración y descripción se han de considerar ilustrativas o ejemplares y no restrictivas; la invención no está limitada a las realizaciones desveladas.

Por ejemplo, en las realizaciones anteriormente descritas, la mayoría de las unidades funcionales del reproductor de vídeo se implementan como software que se ejecuta en un procesador de fin general pero la decodificación de vídeo se realiza mediante un procesador de hardware separado. Esto no es esencial. Por ejemplo, las unidades funcionales tales como la unidad de aleatorización y la unidad de desaleatorización pueden implementarse en hardware o en software. Análogamente, no es esencial que el decodificador de vídeo sea un procesador de vídeo de hardware particular. En otras realizaciones, podría implementarse mediante software que se ejecuta en un procesador de fin general - ya sea el mismo procesador o un procesador diferente en el que se está ejecutando la aplicación de reproductor de vídeo.

En las realizaciones anteriormente descritas, cuando se realiza aleatorización mediante inserción de AU, la AU insertada era una copia de otra AU en el mismo flujo de vídeo. Otra posibilidad es sintetizar una AU artificial. Por ejemplo, la unidad de aleatorización puede generar una AU que representa un fotograma nrB con vectores de movimiento aleatorios y sin textura (es decir, una señal de diferencia de fotograma de movimiento compensado que es cero). Un fotograma de este tipo puede generarse automáticamente basándose en características de alto nivel del flujo (tales como anchura, altura, etc.). Puesto que se codifica usando vectores de movimiento únicamente, después de que se ha decodificado cada bloque del fotograma sintetizado será una copia (aleatoriamente seleccionada) de parte de un fotograma de referencia arbitrario. Este método puede proporcionar mayor molestia visual que simplemente insertar una copia de una AU entera.

Pueden entenderse otras variaciones a las realizaciones desveladas y efectuarse por los expertos en la materia al poner en práctica la invención reivindicada, a partir de un estudio de los dibujos, la divulgación y las reivindicaciones adjuntas. En las reivindicaciones, la palabra “que comprende” no excluye otros elementos o etapas, y el artículo indefinido “un”, “una”, no excluye una pluralidad. Un único procesador u otra unidad pueden satisfacer las funciones de varios elementos indicados en las reivindicaciones. El simple hecho de que se indiquen ciertas medidas en reivindicaciones mutuamente dependientes diferentes no indica que una combinación de estas medidas no pueda usarse para aprovechamiento. Un programa informático puede almacenarse y/o distribuirse en un medio adecuado, tal como un medio de almacenamiento óptico o un medio de estado sólido suministrado junto con o como parte de otro hardware, pero puede distribuirse también en otras formas, tales como mediante internet u otros sistemas de telecomunicación alámbricos o inalámbricos. Cualquier signo de referencia en las reivindicaciones no debería interpretarse como que limita el alcance.

Apéndice - estructuras de datos H.264

Como se ha explicado anteriormente, en el contexto de los ejemplos detallados, un flujo de vídeo comprimido H.264 se aleatoriza modificando el flujo de bits en varios aspectos. Por completitud y referencia conveniente, las definiciones de las estructuras de datos relevantes de la norma H.264 se reproducen a continuación.

La columna a la derecha en cada tabla indica el tamaño de cada campo de datos. El número entre paréntesis es el tamaño, en bits; "v" indica un campo de tamaño variable. Los siguientes tipos se definen en la norma H.264 (sección 7.2):

- u(n): número entero sin signo que usa n bits. Cuando n es "v" en la tabla de sintaxis, el número de bits varía de una manera dependiente del valor de otros elementos de sintaxis. El proceso de análisis para este descriptor se especifica mediante el valor de retorno de la función read_bits(n) interpretado como una representación binaria de un número entero sin signo con el bit más significativo escrito en primer lugar.

- ue(v): elemento de sintaxis codificado con Exp-Golomb de número entero sin signo con el primer bit a la izquierda.

- se(v): elemento de sintaxis codificado con Exp-Golomb de número entero con signo con el primer bit a la izquierda.

Conjunto de parámetros de secuencia (SPS)

Existe al menos una aparición de SPS en cada flujo de vídeo. Sin embargo, SPS puede aparecer más frecuentemente que una vez por vídeo. En particular, para vídeo de difusión, donde el usuario final puede unirse a la sesión en cualquier momento; o en el caso de flujo continuo adaptativo, donde los parámetros de secuencia pueden cambiar dependiendo de las condiciones de la red. Los parámetros encontrados en el SPS son aplicables a todas las AU posteriores hasta que se encuentre un nuevo SPS en el flujo.

seq_parameter_set_data() {	
profile_idc	u(8)
constraint_set0_flag	u(1)
constraint_set1_flag	u(1)
constraint_set2_flag	u(1)
constraint_set3_flag	u(1)
constraint_set4_flag	u(1)
constraint_set5_flag	u(1)
reserved_zero_2bits /* igual a 0 */	u(2)
Level_idc	u(8)
seq_parameter_set_id	ue(v)

if(profile_idc == 100 | | profile_idc == 110 | |
profile_idc == 122 | | profile_idc == 244 | | profile_idc == 44 | |
profile_idc == 83 | | profile_idc == 86 | | profile_idc == 118 | |
profile_idc == 128) {	

chroma_format_ idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
qpprime_y_zero_transform_ bypass_flag	u(1)
seq_scaling_matrix_present_flag	u(1)
if(seq_scaling_matrix_present_flag)	
for(i = 0; i < ((chroma_format_idc != 3) ? 8 : 12); i++) {	
seq_scaling_list_present_flag[i]	u(1)
if(seq_scaling_list_present_flag[i])	
if(i<6)	
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	
else	
scaling_list(ScalingList8x8[i - 6], 64, UseDefaultScalingMatrix8x8Flag[i - 6])	
}	
}	
log2_max_frame_num_minus4	ue(v)

seq_parameter_set_data() {	
pic_order_cnt_type	ue(v)
if(pic_order_cnt_type == 0)	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
else if(pic_order_cnt_type == 1) {	
delta_pic_order_always_zero_flag	u(1)
offset_for_non_ref_ pic	se(v)
offset_for_top_to_bottom_field	se(v)
num_ref_frames_in_pic_order_cnt_cycle	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)	
offset_for_ ref_frame [i]	se(v)
}	
max_num_ref_frames	ue(v)
gaps_in_frame_num_value_allowed_flag	u(1)
pic_width_in_mbs_minus1	ue(v)
pic_height_in_map_units_minus1	ue(v)
frame_mbs_only_flag	u(1)
if(!frame_mbs_only_flag)	
mb_adaptive_frame_field_flag	u(1)
direct_8x8_inference_flag	u(1)
frame_cropping_flag	us)
if(frame_cropping_flag) {	
frame_crop_left_offset	ue(v)
frame_crop_right_offset	ue(v)
frame_crop_top_offset	ue(v)
frame_crop_bottom_offset	ue(v)
}	
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters()	
}	

Conjunto de parámetros de instantánea (PPS)

Hay al menos una aparición de PPS en cada flujo de vídeo. Sin embargo, PPS puede aparecer más frecuentemente que una vez por vídeo. En particular, para vídeo de difusión, donde el usuario final puede unirse a la sesión en cualquier momento, o en el caso de flujo continuo adaptativo, donde los parámetros de secuencia pueden cambiar dependiendo de las condiciones de la red. Los parámetros encontrados en el PPS son aplicables a todas las AU posteriores hasta que se encuentre un nuevo PPS en el flujo. SPS y PPS se insertan normalmente en el flujo uno después del otro.

pic_parameter_set_rbsp() {	
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag	u(1)
bottom_field_pic_order_in_frame_present_flag	u(1)
num_siice_groups_minus1	ue(v)
if(num_slice_groups_minus1 > 0) {	
slice_group_map_type	ue(v)
if(slice_group_map_type == 0)	
for(iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++)	
run_length_minus1[iGroup]	ue(v)
else if(slice_group_map_type == 2)	
for(iGroup = 0; iGroup < num_slice_groups_minus1; iGroup++) {	
top_left[iGroup]	ue(v)
bottom_right[iGroup]	ue(v)
}	
else if(slice_group_map_type == 3 || slice_group_map_type == 4 || slice_group_map_type == 5) {	
slice_group_change_direction_flag	u(1)

pic_parameter_set_rbsp() {	
slice_group_change_rate_minus1	ue(v)
} else if(slice_group_map_type == 6) {	
pic_size_in_map_units_minus1	ue(v)
for(i = 0; i <= pic_size_in_map_units_minus1; i++)	
slice_group_id[i]	u(v)
}	
}	
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
pic_init_qp_minus26 /* relativo a 26 */	se(v)
pic_init_qs_minus26 /* relativo a 26 */	se(v)
chroma_qp_index_offset	se(v)
deblocking_filter_control_present_flag	u(1)
constrained_intra_pred_flag	u(1)
redundant_pic_cnt_present_flag	u(1)
if(more_rbsp_data()) {	
transform_ 8x8_ mode_flag	u(1)
pic_scaling_matrix_present_flag	u(1)
if(pic_scaling_matrix_present_flag)	
for(i=0; i < 6 + ((chroma_format_idc != 3) ? 2 : 6) * transform_8x8_mode_flag; i++) {	
pic_scaling_list_present_flag[i]	u(1)
if(pic_scaling_list_present_flag[i])	
if(i<6)	
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	
else	
scaling_list(ScalingList8x8[i-6], 64, UseDefaultScalingMatrix8x8Flag[i-6])	
}	
second_chroma_qp_index_offset	se(v)
}	
rbsp_trailing_bits()	
}	

Sintaxis de encabezamiento de recorte

Típicamente, hay un corte por AU que representa un fotograma de vídeo. Sin embargo, la norma no evita que un codificador cree muchos cortes por AU, representando cada corte una porción de un fotograma de vídeo. Para cada corte, hay un encabezamiento de corte en el flujo de vídeo.

slice_header() {	
first_mb_in_slice	ue(v)
slice_type	ue(v)
pic_parameter_set_id	ue(v)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
frame_num	u(v)
if(!frame_mbs_only_flag) {	
field_pic_flag	u(1)
if(field_pic_flag)	
bottom_field_flag	u(1)
}	
if(IdrPicFlag)	
idr_pic_id	ue(v)
if(pic_order_cnt_type == 0) {	
pic_order_cnt_lsb	u(v)

slice_header() {	
if(bottom_field_pic_order_in_frame_present_flag && !field_pic_flag)	
delta_pic_order_cnt_bottom	se(v)
}	
if(pic_order_cnt_type == 1 && !delta_pic_order_always_zero_flag) {	
delta_pic_order_cnt[0]	se(v)
if(bottom_field_pic_order_in_frame_present_flag && !field_pic_flag)	
delta_pic_order_cnt[1]	se(v)
}	
if(redundantpiccnt_presentflag)	
redundant_pic_cnt	ue(v)
if(slice_type == B)	
direct-spatial_mv_pred_flag	u(1)
if(slice_type == P | | slice_type == SP | | slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
if(nal_unit_type == 20)	
ref_pic_list_mvc_modification() /* véase Anexo H */	
else	
ref_pic_list_modification()	
if((weighted_pred_flag && (slice_type == P | | slice_type == SP)) | |(weighted_bipred_idc == 1 && slice_type == B))	
pred_weight_table()	
if(nal_ref_idc != 0)	
dec_ref_pic_marking()	
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)	
cabac_init_idc	ue(v)
slice_qp_delta	se(v)
if(slice_type == SP | | slice_type == SI) {	
if(slice_type == SP)	
sp_for_switch_flag	u(1)
slice_qs_delta	se(v)
}	
if(deblocking_filter_control_present_flag) {	
disable_deblocking_filter_idc	ue(v)
if(disable_deblocking_filter_idc != 1) {	
slice_alpha_c0_offset_div2	se(v)
slice_beta_offset_div2	se(v)
}	
}	
if(num_slice_groups_minus1 > 0 && slice_group_map_type >= 3 && slice_group_map_type <= 5)	
slice_group_change_cycle	u(v)
}	

REIVINDICACIONES

1. Aparato para decodificar un vídeo comprimido, comprendiendo el aparato:

una unidad de aleatorización de vídeo (220), adaptada para recibir el vídeo comprimido y aleatorizarlo, para producir un vídeo comprimido aleatorizado, en el que la aleatorización realizada mediante la unidad de aleatorización de vídeo (220) comprende una o más operaciones realizadas en unidades de acceso individuales del vídeo comprimido; y en el que la unidad de aleatorización de vídeo (220) está adaptada para producir metadatos de aleatorización que describen una operación de aleatorización aplicada a las unidades de acceso realizada mediante la unidad de aleatorización de vídeo (220);
un decodificador de vídeo (130), dispuesto para recibir el vídeo comprimido aleatorizado desde la unidad de aleatorización (220) y adaptado para decodificarlo, para producir un vídeo descomprimido aleatorizado en el dominio de píxel; y
una unidad de desaleatorización de vídeo (230), dispuesta para recibir el vídeo descomprimido aleatorizado desde el decodificador de vídeo y adaptada para desaleatorizarlo para producir un vídeo descomprimido desaleatorizado, en el que la unidad de desaleatorización de vídeo (230) está dispuesta para recibir los metadatos de aleatorización desde la unidad de aleatorización de vídeo (220) y adaptada para usarlos para desaleatorizar el vídeo descomprimido aleatorizado en el dominio de píxel recibido desde el decodificador de vídeo (130).

2. El aparato de la reivindicación 1, en el que la unidad de aleatorización de vídeo (220) está adaptada para aleatorizar el vídeo comprimido de una manera que:

no evita la decodificación del vídeo mediante el decodificador de vídeo (130);
modifica el contenido visual del vídeo descomprimido aleatorizado después de decodificación; y
es reversible mediante la unidad de desaleatorización de vídeo (230), de manera que el vídeo descomprimido desaleatorizado es idéntico o sustancialmente idéntico al vídeo que se produciría si el vídeo comprimido se hubiera de decodificar mediante el decodificador de vídeo, sin aleatorización.

3. El aparato de la reivindicación 1 o la reivindicación 2, en el que la aleatorización realizada mediante la unidad de aleatorización de vídeo (220) comprende al menos uno de:

cambiar una ordenación de unidades de acceso en el vídeo comprimido;
insertar una o más unidades de acceso adicionales en el vídeo comprimido; y
eliminar una o más unidades de acceso del vídeo comprimido.

4. El aparato de una cualquiera de las reivindicaciones 1 a 3, en el que la unidad de aleatorización de vídeo (220) está controlada para aplicar aleatorización a primeras porciones del vídeo y no aplicar aleatorización a segundas porciones del vídeo.

5. El aparato de la reivindicación 4, en el que las primeras y segundas porciones se seleccionan aleatoriamente o pseudo-aleatoriamente.

6. El aparato de una cualquiera de las reivindicaciones 1 a 5, en el que el decodificador de vídeo (130) es un decodificador de vídeo de hardware.

7. El aparato de una cualquiera de las reivindicaciones 1 a 6, en el que el vídeo comprimido y el decodificador de vídeo cumplen con una norma de codificación de vídeo predeterminada; y la unidad de aleatorización de vídeo (220) está adaptada para aleatorizar el vídeo comprimido de tal manera que el vídeo comprimido aleatorizado cumple con la misma norma de codificación de vídeo.

8. El aparato de la reivindicación 7, en el que el decodificador de vídeo (130) y el vídeo comprimido cumplen con la norma H.264 y en el que la aleatorización comprende modificar el recuento de orden de instantánea, de una pluralidad de unidades de acceso en el vídeo comprimido H.264.

9. El aparato de la reivindicación 8, en el que la unidad de aleatorización de vídeo (220) está adaptada para:

detectar si el recuento de orden de instantánea está codificado implícitamente en el vídeo comprimido y, en caso afirmativo, en respuesta:

modificar información de encabezamiento del vídeo comprimido para señalizar que el recuento de orden de instantánea está codificado explícitamente; y
asignar un valor de recuento de orden de instantánea explícito en cada unidad de acceso del vídeo comprimido.

10. El aparato de la reivindicación 8 o la reivindicación 9, en el que la unidad de aleatorización de vídeo (220) está adaptada para detectar en el vídeo comprimido un modo de codificación que requeriría que el decodificador de vídeo usara el recuento de orden de instantánea para decodificar una o más unidades de acceso y, en respuesta:

desactivar la aleatorización; o
modificar el vídeo comprimido para convertirlo a otro perfil H.264, de modo que el decodificador de vídeo no necesite usar el recuento de orden de instantánea para decodificar las una o más unidades de acceso.

11. El aparato de una cualquiera de las reivindicaciones 7 a 9 en el que la unidad de aleatorización de vídeo (220) está adaptada para:

detectar si el vídeo comprimido está codificado de acuerdo con el perfil de línea de base de la norma H.264 y, en caso afirmativo, en respuesta,
modificar el vídeo comprimido para convertirlo a otro perfil H.264.

12. Un método de decodificación de un vídeo comprimido, que comprende:

aleatorizar el vídeo comprimido, para producir un vídeo comprimido aleatorizado; en el que la aleatorización realizada comprende una o más operaciones realizadas en unidades de acceso individuales del vídeo comprimido; y en el que se producen metadatos de aleatorización de vídeo que describen una operación de aleatorización aplicada a las unidades de acceso realizada mediante una unidad de aleatorización de vídeo (220);
entregar el vídeo comprimido aleatorizado a un decodificador de vídeo (130), para decodificar el vídeo comprimido aleatorizado para producir un vídeo descomprimido aleatorizado en el dominio de píxel;
recibir desde el decodificador de vídeo el vídeo descomprimido aleatorizado; y
desaleatorizar el vídeo descomprimido aleatorizado, para producir un vídeo descomprimido desaleatorizado, en el que los metadatos de aleatorización se reciben desde la unidad de aleatorización de vídeo (220) para desaleatorizar el vídeo descomprimido aleatorizado en el dominio de píxel recibido desde el decodificador de vídeo (130).

13. Un medio legible por ordenador no transitorio que almacena un programa informático, comprendiendo el programa informático medios de código de programa informático adaptados para realizar todas las etapas de la reivindicación 12 cuando dicho programa se ejecuta en un ordenador.

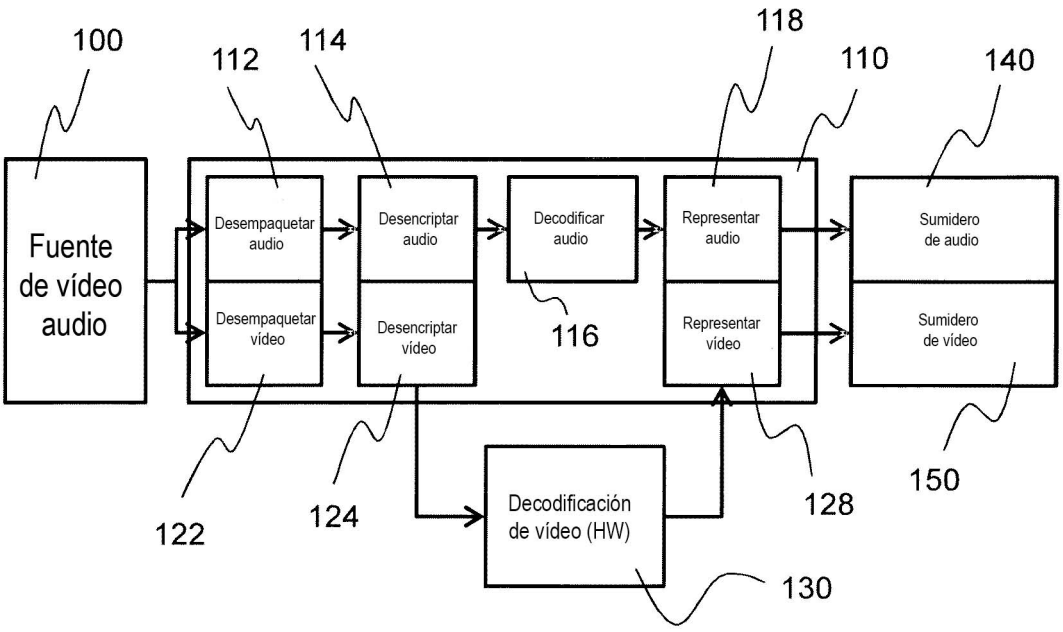
100
112
114
118
110
140
Fuente de vídeo audio
Desempaquetar audio
Desempaquetar vídeo
Desencriptar audio
Desencriptar vídeo
Decodificar audio
116
Representar audio
Representar vídeo
Sumidero de audio
Sumidero de vídeo
122
124
Decodificación de vídeo (HW)
128
150
130

FIG. 1

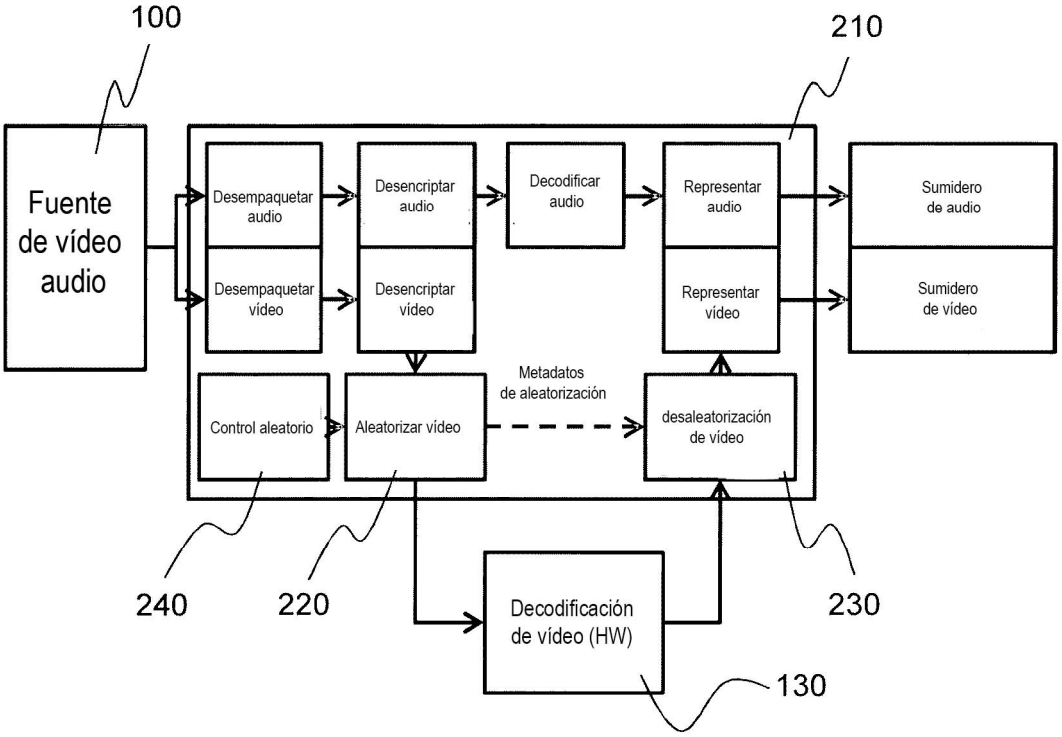
100
210
Fuente de vídeo audio
Desempaquetar audio
Desempaquetar vídeo
Desencriptar audio
Desencriptar vídeo
Decodificar audio
Representar audio
Representar vídeo
Sumidero de audio
Sumidero de vídeo
Metadatos de aleatorización
Control aleatorio
Aleatorizar vídeo
desaleatorización de vídeo
240
220
Decodificación de vídeo (HW)
230
130

FIG. 2

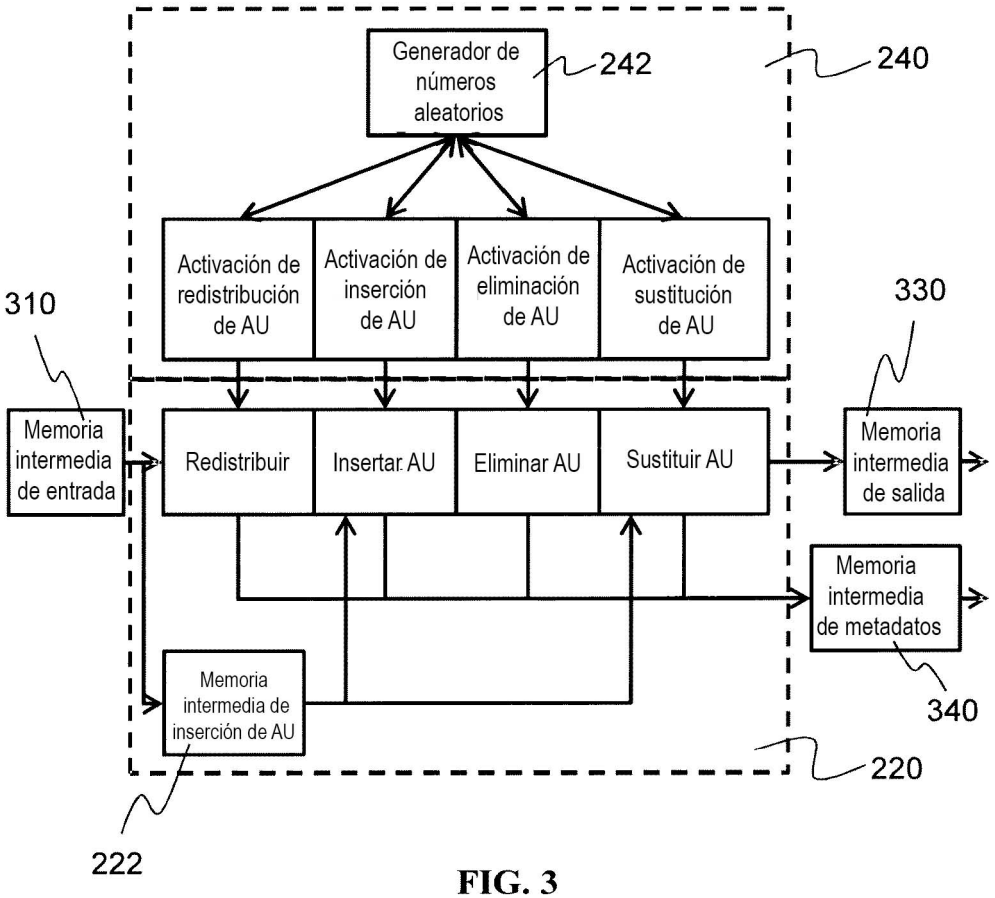
Generador de números aleatorios
242
240
Activación de redistribución de AU
Activación de inserción de AU
Activación de eliminación de AU
Activación de sustitución de AU
310
330
Memoria intermedia de entrada
Redistribuir
Insertar AU
Eliminar AU
Sustituir AU
Memoria intermedia de salida
Memoria intermedia de metadatos
340
Memoria intermedia de inserción de AU
220
222

FIG. 3

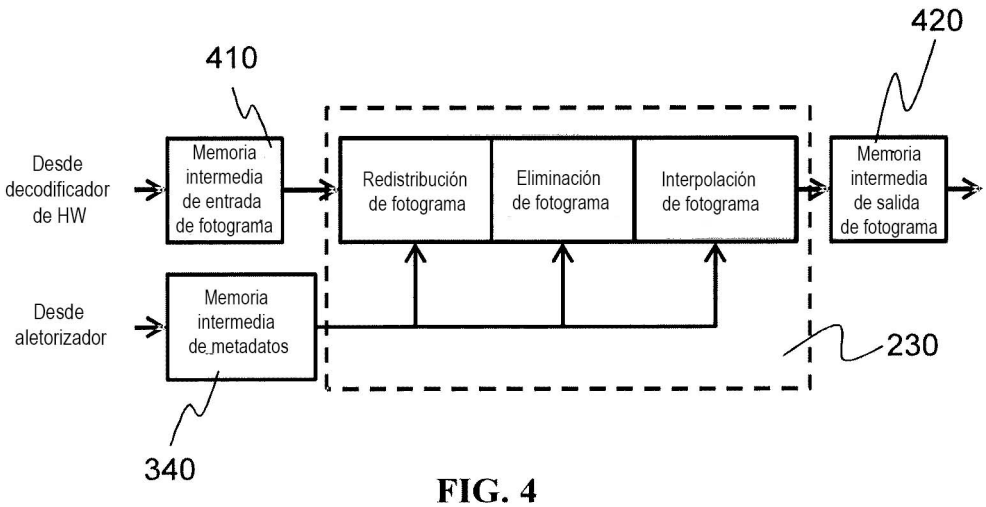
420
410
Desde decodificador de HW
Memoria intermedia de entrada de fotograma
Redistribución de fotograma
Eliminación de fotograma
Interpolación de fotograma
Memoria intermedia de salida de fotograma
Desde aletorizador
Memoria intermedia de metadatos
230
340

FIG. 4

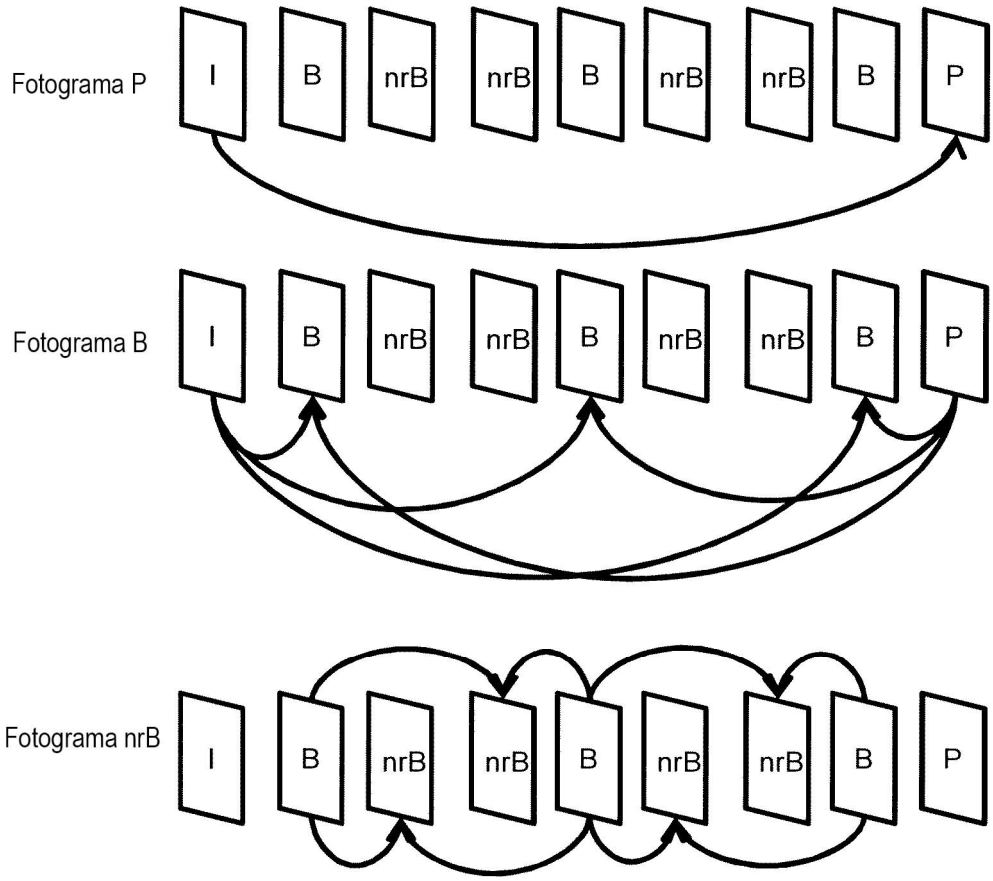
Fotograma P
I
B
nrB
nrB
B
nrB
nrB
B
P
Fotograma B
I
B
nrB
nrB
B
nrB
nrB
B
P
Fotograma nrB
I
B
nrB
nrB
B
nrB
nrB
B
P

FIG. 5

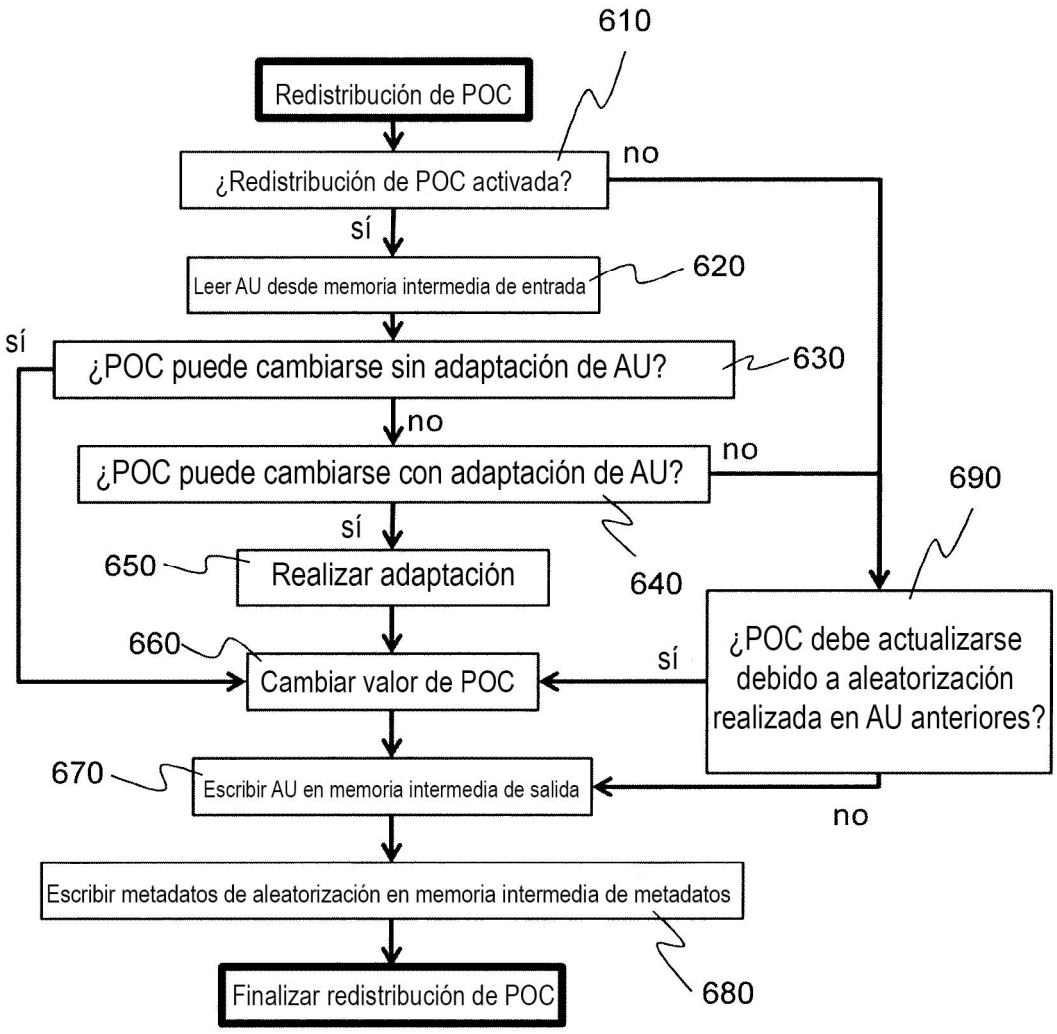
610
Redistribución de POC
¿Redistribución de POC activada?
no
sí
Leer AU desde memoria intermedia de entrada
620
sí
¿POC puede cambiarse sin adaptación de AU?
630
no
¿POC puede cambiarse con adaptación de AU?
no
690
sí
650
Realizar adaptación
640
660
Cambiar valor de POC
sí
¿POC debe actualizarse debido a aleatorización realizada en AU anteriores?
670
Escribir AU en memoria intermedia de salida
no
Escribir metadatos de aleatorización en memoria intermedia de metadatos
680
Finalizar redistribución de POC

FIG. 6

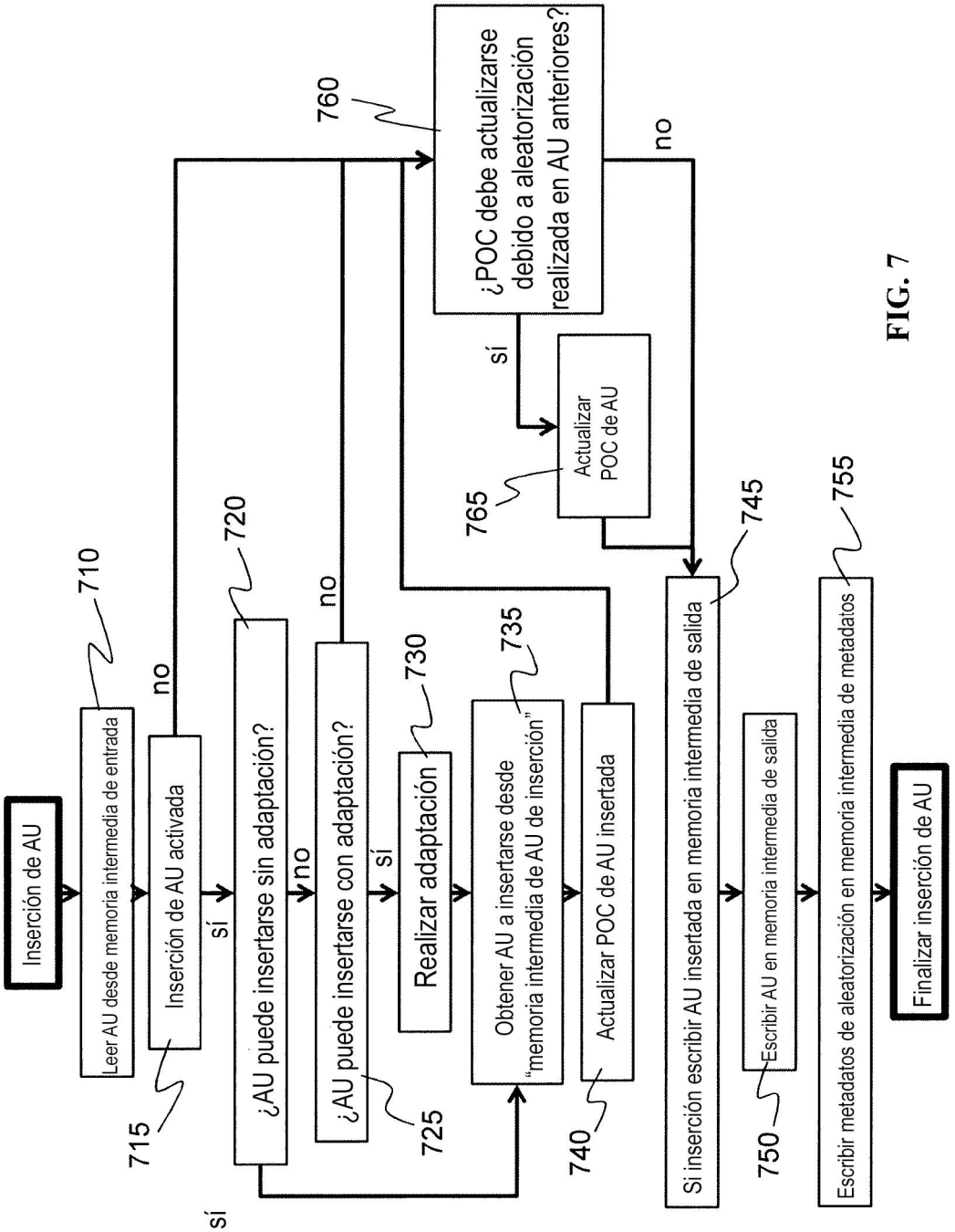
Inserción de AU
Leer AU desde memoria intermedia de entrada
710
Inserción de AU activada
715
no
sí
¿AU puede insertarse sin adaptación?
720
no
sí
¿AU puede insertarse con adaptación?
725
no
sí
Realizar adaptación
730
Obtener AU a insertarse desde "memoria intermedia de AU de inserción"
735
Actualizar POC de AU insertada
740
¿POC debe actualizarse debido a aleatorización realizada en AU anteriores?
760
sí
no
Actualizar POC de AU
765
Si inserción escribir AU insertada en memoria intermedia de salida
745
Escribir AU en memoria intermedia de salida
750
Escribir metadatos de aleatorización en memoria intermedia de metadatos
755
Finalizar inserción de AU

FIG. 7

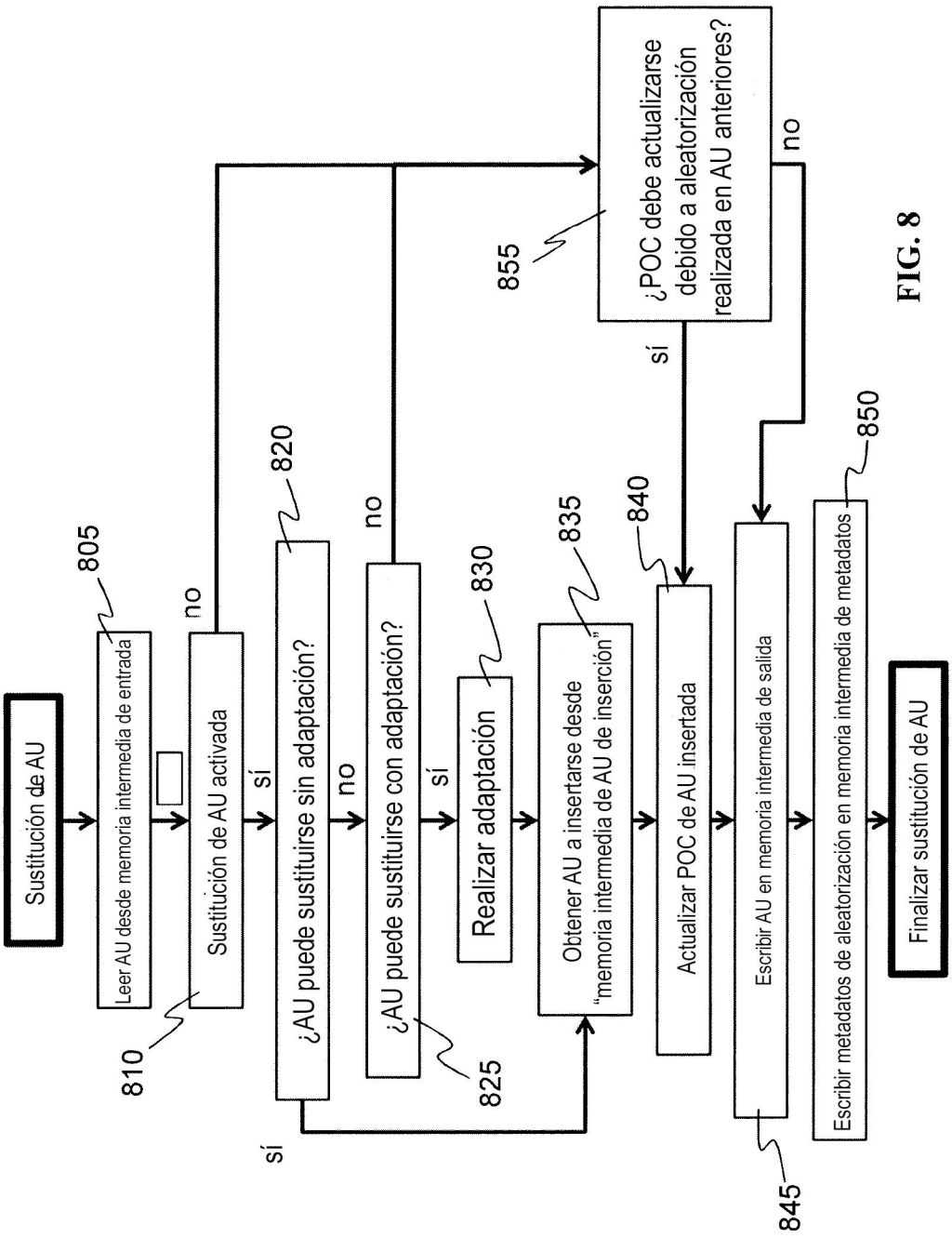
Sustitución de AU
805
Leer AU desde memoria intermedia de entrada
810
Sustitución de AU activada
no
sí
820
¿AU puede sustituirse sin adaptación?
no
sí
825
¿AU puede sustituirse con adaptación?
no
sí
830
Realizar adaptación
835
Obtener AU a insertarse desde "memoria intermedia de AU de inserción"
840
Actualizar POC de AU insertada
845
Escribir AU en memoria intermedia de salida
850
Escribir metadatos de aleatorización en memoria intermedia de metadatos
Finalizar sustitución de AU
855
¿POC debe actualizarse debido a aleatorización realizada en AU anteriores?
sí
no

FIG. 8

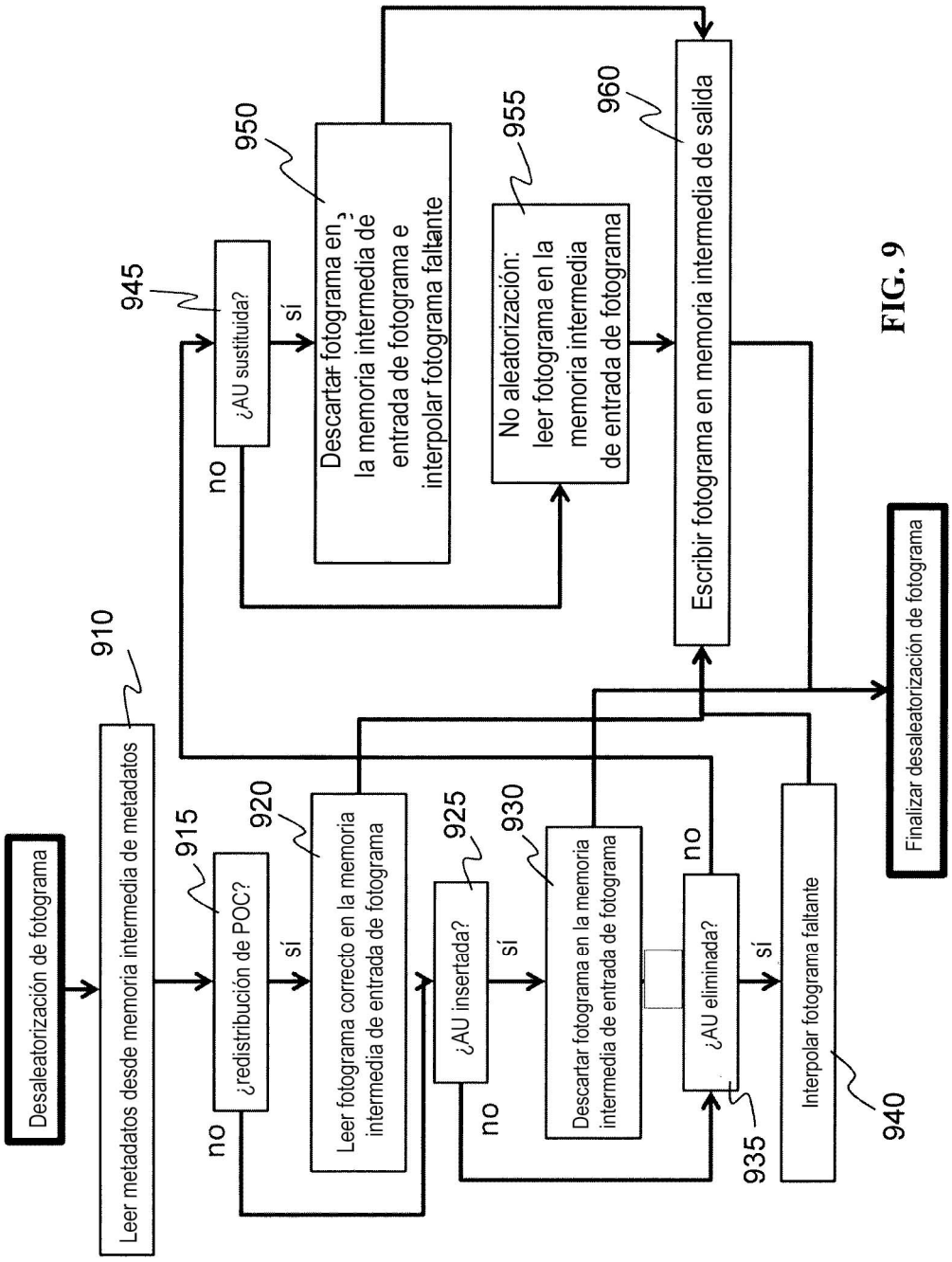
Desaleatorización de fotograma
Leer metadatos desde memoria intermedia de metadatos
910
¿redistribución de POC?
915
sí
no
Leer fotograma correcto en la memoria intermedia de entrada de fotograma
920
¿AU insertada?
925
sí
no
Descartar fotograma en la memoria intermedia de entrada de fotograma
930
¿AU eliminada?
935
no
sí
Interpolar fotograma faltante
940
¿AU sustituida?
945
sí
no
Descartar fotograma en la memoria intermedia de entrada de fotograma e interpolar fotograma faltante
950
No aleatorización: leer fotograma en la memoria intermedia de entrada de fotograma
955
Escribir fotograma en memoria intermedia de salida
960
Finalizar desaleatorización de fotograma

FIG. 9

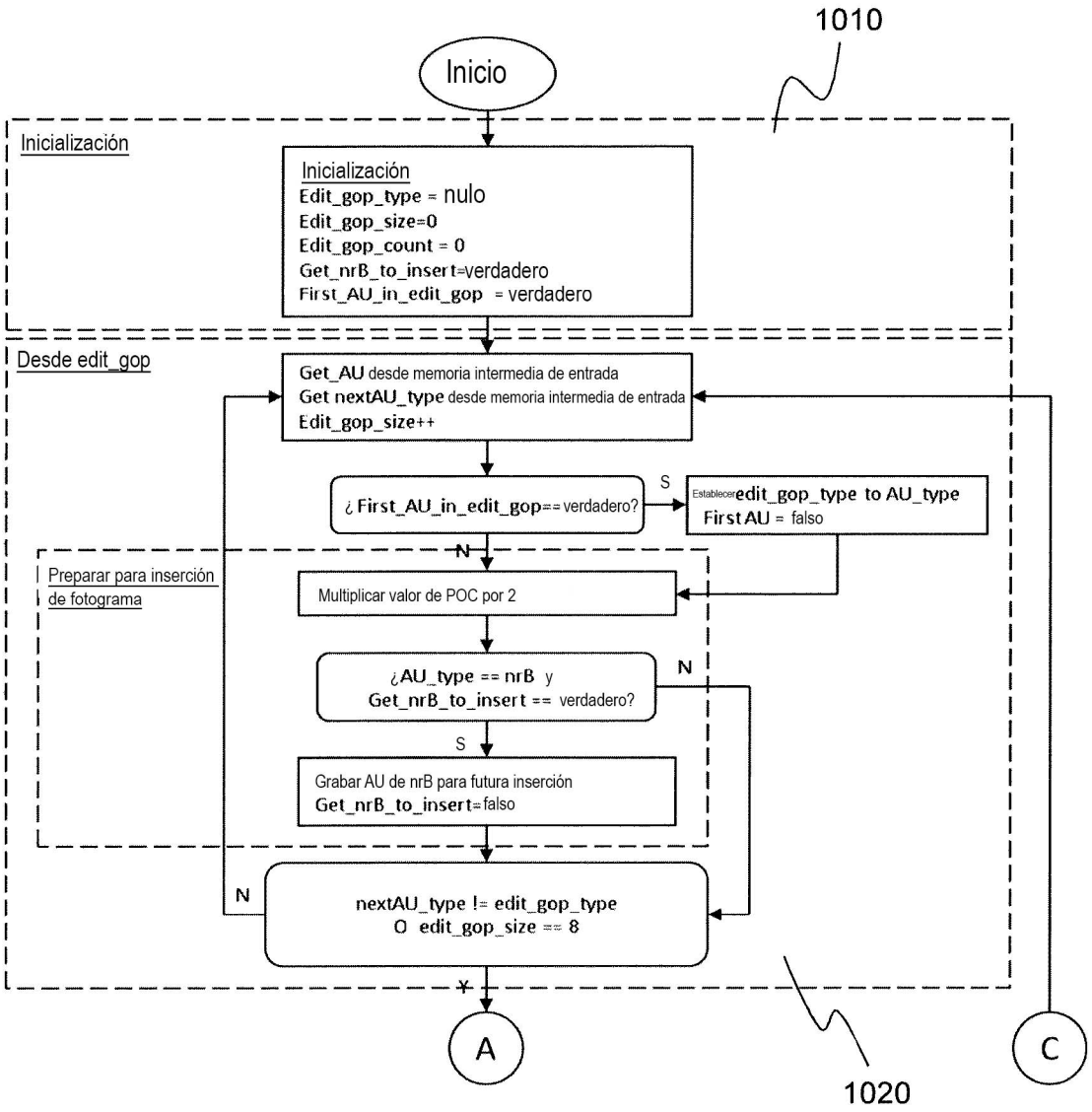
1010
Inicio
Inicialización
Inicialización
Edit_gop_type = nulo
Edit_gop_size=0
Edit_gop_count = 0
Get_nrB_to_insert=verdadero
First_AU_in_edit_gop = verdadero
Desde edit_gop
Get_AU desde memoria intermedia de entrada
Get nextAU_type desde memoria intermedia de entrada
Edit_gop_size++
¿ First_AU_in_edit_gop== verdadero?
S
Establecer edit_gop_type to AU_type
First AU = falso
N
Preparar para inserción de fotograma
Multiplicar valor de POC por 2
¿AU_type == nrB y
Get_nrB_to_insert == verdadero?
N
S
Grabar AU de nrB para futura inserción
Get_nrB_to_insert=falso
N
nextAU_type != edit_gop_type
O edit_gop_size == 8
Y
A
C
1020

FIG. 10

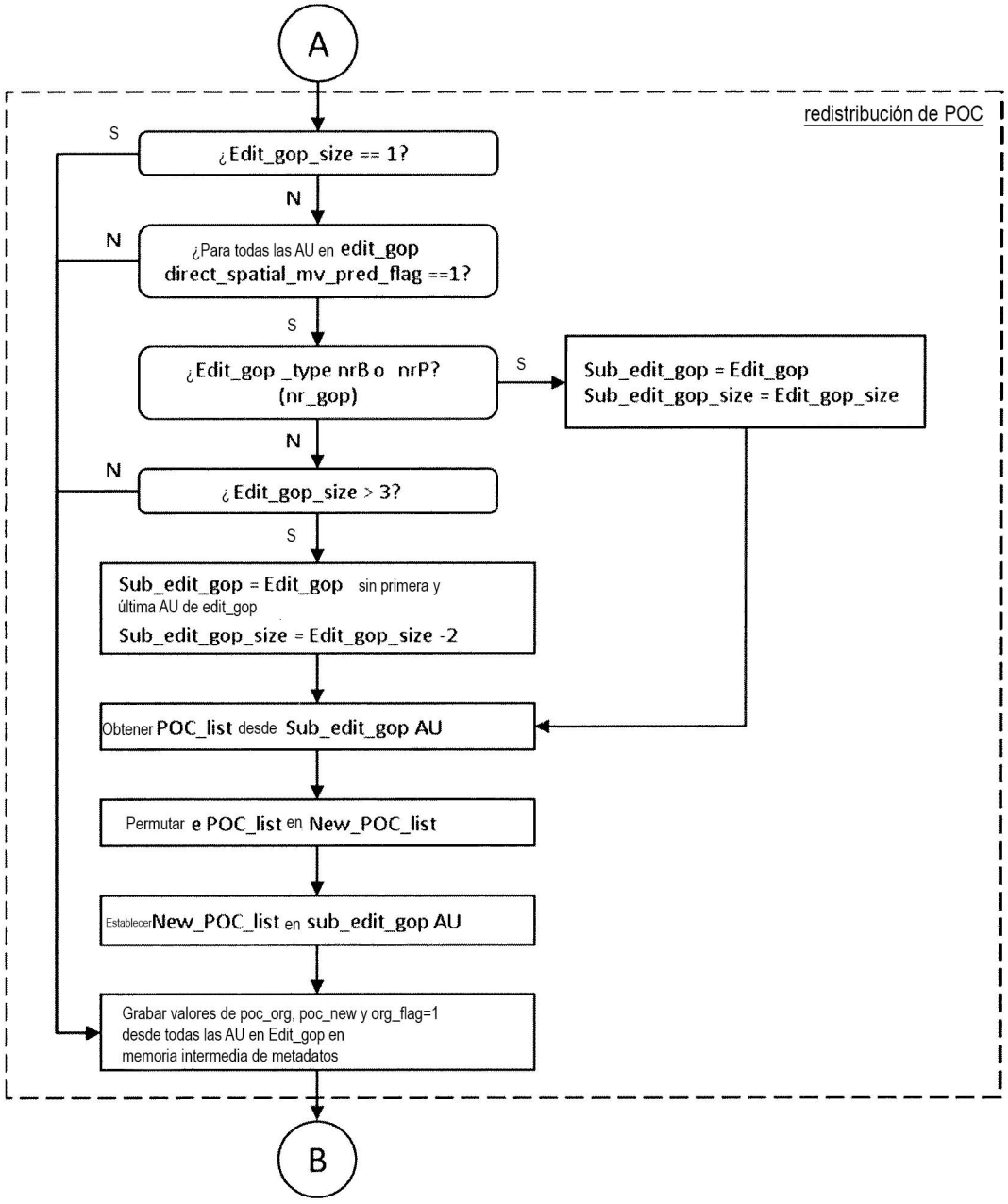
A
redistribución de POC
S
¿Edit_gop_size == 1?
N
N
¿Para todas las AU en edit_gop
direct_spatial_mv_pred_flag ==1?
S
¿Edit_gop _type nrB o nrP?
(nr_gop)
S
Sub_edit_gop = Edit_gop
Sub_edit_gop_size = Edit_gop_size
N
N
¿ Edit_gop_size > 3?
S
Sub_edit_gop = Edit_gop sin primera y
última AU de edit_gop
Sub_edit_gop_size = Edit_gop_size -2
Obtener POC_list desde Sub_edit_gop AU
Permutar e POC_list en New_POC_list
EstablecerNew_POC_list en sub_edit_gop AU
Grabar valores de poc_org, poc_new y org_flag=1
desde todas las AU en Edit_gop en
memoria intermedia de metadatos
B

FIG. 11

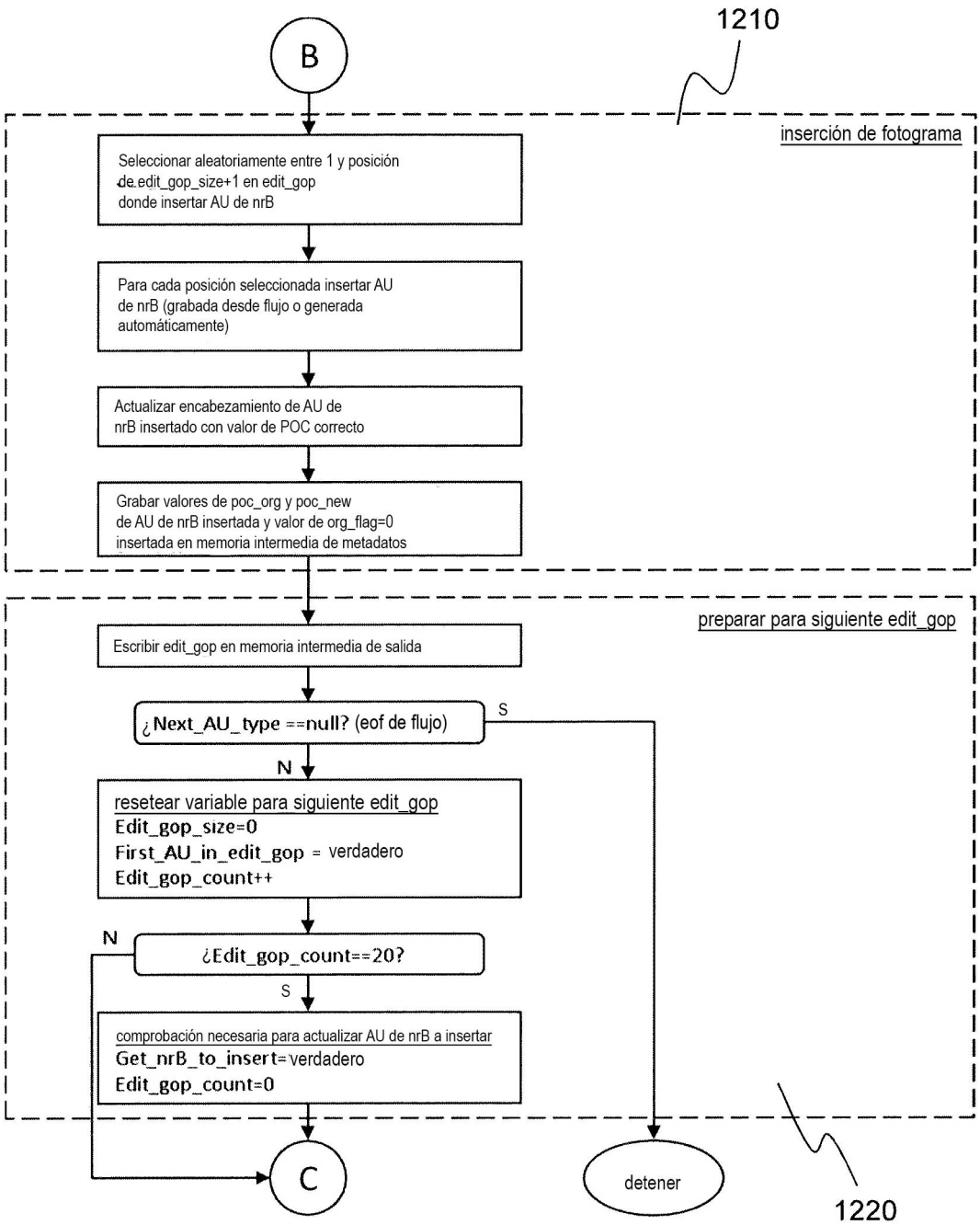
1210
B
inserción de fotograma
Seleccionar aleatoriamente entre 1 y posición de edit_gop_size+1 en edit_gop donde insertar AU de nrB
Para cada posición seleccionada insertar AU de nrB (grabada desde flujo o generada automáticamente)
Actualizar encabezamiento de AU de nrB insertado con valor de POC correcto
Grabar valores de poc_org y poc_new de AU de nrB insertada y valor de org_flag=0 insertada en memoria intermedia de metadatos
preparar para siguiente edit_gop
Escribir edit_gop en memoria intermedia de salida
¿Next_AU_type ==null? (eof de flujo)
S
N
resetear variable para siguiente edit_gop
Edit_gop_size=0
First_AU_in_edit_gop = verdadero
Edit_gop_count++
N
¿Edit_gop_count==20?
S
comprobación necesaria para actualizar AU de nrB a insertar
Get_nrB_to_insert= verdadero
Edit_gop_count=0
C
detener
1220

FIG. 12

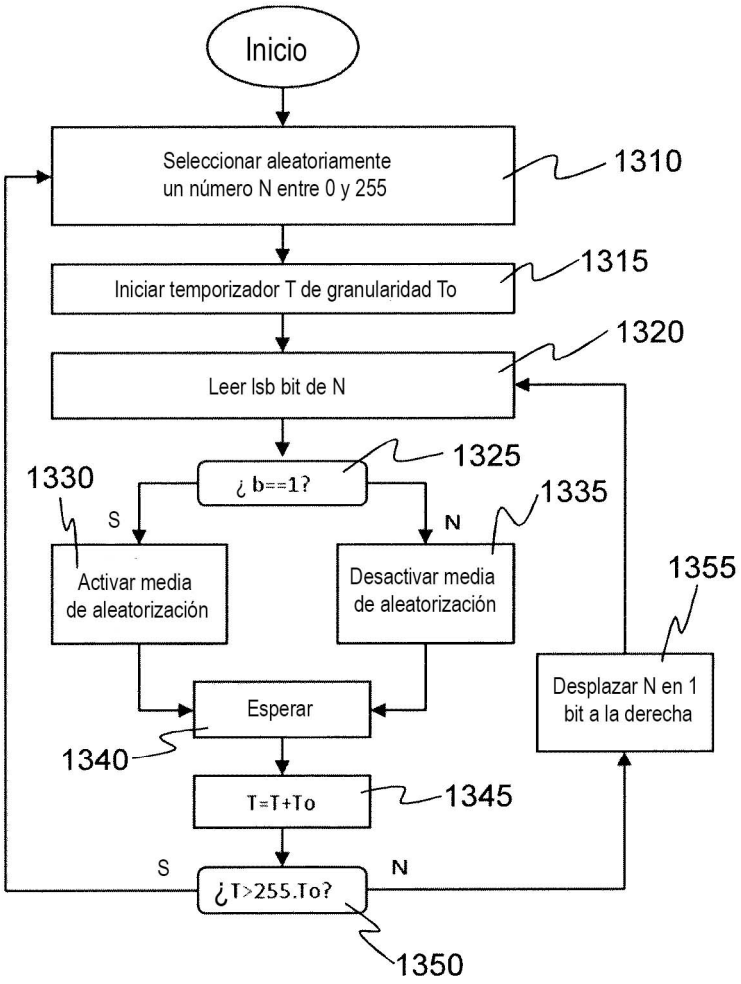
Inicio
Seleccionar aleatoriamente un número N entre 0 y 255
1310
Iniciar temporizador T de granularidad To
1315
Leer lsb bit de N
1320
¿ b==1?
1325
1330
S
N
1335
Activar media de aleatorización
Desactivar media de aleatorización
1355
Desplazar N en 1 bit a la derecha
Esperar
1340
T=T+To
1345
S
N
¿T>255.To?
1350

FIG. 13