(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **3 001 562**

(51) Int. Cl.:

H04N 21/4728 (2011.01)
H04N 21/84 (2011.01)
H04N 21/854 (2011.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **09.02.2016 E 22187970 (3)**

(97) Fecha y número de publicación de la concesión europea: **25.12.2024 EP 4102850**

(54) Título: **Encapsulación de datos de imágenes**

(30) Prioridad:

10.02.2015 GB 201502205
17.02.2015 GB 201502666

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
05.03.2025

(73) Titular/es:

CANON KABUSHIKI KAISHA (100.00%)
30-2, Shimomaruko 3-chome Ohta-ku
Tokyo, Tokyo 146-8501, JP

(72) Inventor/es:

MAZE, FRÉDÉRIC;
DENOUAL, FRANCK;
CONCOLATO, CYRIL y
LE FEUVRE, JEAN

(74) Agente/Representante:

DURAN-CORRETJER, S.L.P

DESCRIPCIÓN

Encapsulación de datos de imágenes

SECTOR DE LA INVENCIÓN

La presente invención se refiere al almacenamiento de datos de imágenes, tales como imágenes fijas, ráfagas de imágenes fijas o datos de vídeo en un contenedor multimedia con metadatos descriptivos. Dichos metadatos, en general, proporcionan un acceso fácil a los datos de imágenes y a porciones de los datos de imágenes.

ESTADO DE LA TÉCNICA ANTERIOR

Algunos de los enfoques descritos en esta sección podrían ser implementados pero, no son necesariamente enfoques que hayan sido concebidos o implementados anteriormente. Por lo tanto, los enfoques descritos en esta sección no son necesariamente técnica anterior a las reivindicaciones de esta solicitud, y no están admitidos como técnica anterior por inclusión en esta sección.

El estándar HEVC define un perfil para la codificación de imágenes fijas, y describe herramientas específicas para comprimir imágenes fijas individuales o ráfagas de imágenes fijas. Se ha propuesto una extensión del formato de archivo multimedia basado en ISO (ISO Base Media File Format, ISOBMFF) utilizado para este tipo de datos de imagen para su inclusión en el estándar ISO/IEC 23009, en la Parte 12, bajo el nombre de "Image File Format". El estándar abarca dos formas de almacenamiento correspondientes a diferentes casos de uso:

- el almacenamiento de secuencias de imágenes, con temporización que se utiliza opcionalmente en el decodificador, y en el que las imágenes pueden depender de otras imágenes, y
- el almacenamiento de imágenes individuales y conjuntos de imágenes codificadas de manera independiente.

En el primer caso, la encapsulación es parecida a la encapsulación de las pistas de vídeo en el formato de archivo multimedia basado en ISO (véase el documento "Information technology - Coding of audio-visual objects - Part 12: ISO base media file format", ISO/IEC 14496-12:2008, Tercera edición, octubre de 2008), y se utilizan las mismas herramientas y conceptos, tales como las cajas "trak" y el agrupamiento de muestras para la descripción. La caja "pista" es una caja de formato de archivo que contiene sub-cajas para describir una pista, es decir, una secuencia temporizada de muestras relacionadas.

En el segundo caso, un conjunto de cajas de ISOBMFF, se utilizan las cajas "meta". Estas cajas y su jerarquía ofrecen menos herramientas de descripción que las cajas "pista" y se refieren a "elementos de información" o "elementos", en lugar de a muestras relacionadas.

El formato de archivo de imagen se puede utilizar para visualizar localmente archivos multimedia o para transmitir en continuo presentaciones multimedia. Las imágenes fijas de HEVC tienen muchas aplicaciones que plantean muchos problemas.

Las ráfagas de imágenes son una aplicación. Las ráfagas de imágenes son secuencias de imágenes fijas capturadas por una cámara y almacenadas como una sola representación (muchos elementos de imagen que hacen referencia a un bloque de datos). Los usuarios pueden querer realizar varios tipos de acciones en estas imágenes: seleccionar una como miniatura o portada, aplicar efectos a estas imágenes o similares.

Por lo tanto, existe una necesidad de metadatos descriptivos para identificar la lista de imágenes con sus bytes correspondientes en el bloque de datos.

La fotografía digital es otra aplicación. En la fotografía digital, los usuarios tienen acceso a diferentes resoluciones de la misma imagen (diferentes exposiciones, diferentes enfoques, etc.). Estas diferentes resoluciones tienen que ser almacenadas como metadatos para que se pueda seleccionar una y se pueda localizar y extraer el fragmento de datos correspondiente para su procesamiento (renderizado, edición, transmisión o similar).

Con el aumento de la resolución de la imagen en términos de tamaño, existe la necesidad de proporcionar una descripción suficiente para que solo algunas partes espaciales de estas imágenes grandes puedan ser identificadas y extraídas fácilmente.

Otro tipo de aplicaciones es el acceso a imágenes específicas a partir de una secuencia de vídeo, por ejemplo, para el resumen de vídeo, imágenes de prueba en datos de videovigilancia o similares.

Para este tipo de aplicaciones, existe la necesidad de metadatos de imagen que permitan acceder fácilmente a las imágenes clave, además de los datos de vídeo comprimidos y los metadatos de las pistas de vídeo.

Además, las cámaras profesionales han alcanzado altas resoluciones espaciales. Los vídeos o imágenes con resolución 4K2K son actualmente comunes. Incluso los vídeos o imágenes 8k4k son actualmente comunes. En paralelo, los vídeos se reproducen cada vez más en dispositivos móviles y conectados con capacidades de transmisión de vídeo en continuo. Por lo tanto, dividir los vídeos en teselas resulta importante si el usuario de un dispositivo móvil desea visualizar o desea centrarse en subpartes del vídeo manteniendo o incluso mejorando la calidad. Utilizando teselas, el usuario puede solicitar de manera interactiva subpartes espaciales del vídeo.

Por lo tanto, existe la necesidad de describir estas subpartes espaciales del vídeo de manera compacta en el formato de archivo, para que sean accesibles sin un procesamiento adicional más allá del simple análisis sintáctico de las cajas de metadatos. En el caso de las imágenes correspondientes a los vídeos descritos de esta manera, también resulta de interés para el usuario acceder a subpartes espaciales.

El estándar ISO/IEC 23009 abarca dos modos de encapsular imágenes fijas en el formato de archivo, que han sido analizados recientemente.

Un modo se basa en cajas “pista” y en la noción de secuencia temporizada de muestras relacionadas con herramientas de descripción asociadas, y otro se basa en cajas “meta”, basadas en elementos de información, en lugar de muestras, que proporcionan menos herramientas de descripción, especialmente para la descripción de la zona de interés y el soporte de la teselación.

Por lo tanto, existe la necesidad de proporcionar soporte para la teselación en el nuevo formato de archivo de imagen.

La utilización de teselas es conocida comúnmente en la técnica anterior, especialmente en el momento de la compresión. En lo que respecta a su indexación en el formato de archivo multimedia basado en ISO, existen descriptores de teselación en los borradores para enmienda de la Parte 15 del estándar ISO/IEC 14496 “Carriage of NAL unit structured video in the ISO Base Media File Format”.

Sin embargo, estos descriptores se basan en cajas “pista” y en herramientas de agrupamiento de muestras, y no se pueden utilizar en el formato de archivo de imagen fija cuando se utiliza el enfoque basado en “meta”. Sin dichos descriptores, resulta complicado seleccionar y extraer teselas de una imagen codificada almacenada en este formato de archivo.

La **figura 1** muestra la descripción de una imagen fija codificada con teselas en la caja “meta” (100) del formato de archivo multimedia basado en ISO, tal como se da a conocer en la contribución de MPEG m32254.

Un elemento de información está definido para la imagen completa 101, además de los elementos de información respectivos para cada imagen de tesela (102, 103, 104 y 105). Esos elementos de información se almacenan en una caja denominada “ItemInfoBox” (iinf). La caja (106), denominada “ItemReferenceBox”, del estándar ISO BMFF se utiliza para indicar que existe una relación de “tesela” (107) entre el elemento de información de la imagen completa y los cuatro elementos de información correspondientes a las imágenes de tesela (108). Se utilizan identificadores de cada elemento de información, de modo que una caja (109), denominada “ItemLocationBox”, proporciona el o los intervalos de bytes en los datos codificados (110) que representan cada elemento de información. Otra caja “ItemReferenceBox” (112) se utiliza para asociar metadatos EXIF (111) con el elemento de información para la imagen completa (101) y se crea un bloque de datos (111) correspondiente en la caja de datos multimedia (110). Asimismo, se crea un elemento de información (113) adicional para identificar los metadatos EXIF.

Incluso si la imagen completa y sus teselas se introducen como elementos de información, en este caso no se proporciona información de teselación. Además, al asociar metadatos adicionales con un elemento de información (tal como EXIF), no se crea ningún bloque de datos referenciado mediante un “ItemReferenceBox” adicional.

La reutilización de la información sobre teselación desde EXIF y la reutilización del mecanismo definido en el borrador del formato de archivo de imagen fijas no permitiría describir una cuadrícula no regular con las etiquetas de EXIF existentes.

Por lo tanto, sigue existiendo la necesidad de mejoras en el formato de archivo para imágenes fijas, especialmente imágenes fijas de HEVC. En concreto, existe una necesidad de procedimientos para extraer una zona de interés en imágenes fijas almacenadas con este formato de archivo.

La invención se encuentra dentro del contexto anterior.

CARACTERÍSTICAS DE LA INVENCIÓN

Según la invención, se da a conocer un procedimiento para encapsular un flujo de bits codificado que representa una o varias imágenes, un procedimiento para procesar un archivo de datos encapsulado, un dispositivo servidor, un dispositivo cliente y un programa informático según las reivindicaciones adjuntas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Otras características y ventajas de la invención resultarán evidentes a partir de la siguiente descripción de realizaciones a modo de ejemplo no limitativas, haciendo referencia a los dibujos adjuntos, en los que, además de la figura 1:

- la **figura 2** muestra un ejemplo de un vídeo teselado;
- la **figura 3** muestra diversas configuraciones de tesela/corte en HEVC;
- la **figura 4** muestra la encapsulación de teselas según el formato de archivo multimedia basado en ISO con cajas "pista";
- la **figura 5** muestra los metadatos estándar para describir elementos de información en cajas "meta" del ISOBMFF;
- la **figura 6** muestra una extensión a modo de ejemplo de la descripción del elemento de información;
- la **figura 7** muestra los mecanismos de referencia entre elementos de información;
- la **figura 8** muestra un contexto de implementación de realizaciones de la invención;
- la **figura 9** es un diagrama de bloques esquemático de un dispositivo informático para la implementación de una o varias realizaciones de la invención.

DESCRIPCIÓN DETALLADA DE LA INVENCIÓN

A continuación se describen realizaciones de la invención.

Para comprender mejor el contexto técnico, se explica la teselación de vídeo haciendo referencia a la **figura 2**, que muestra un vídeo (200) que tiene cuadros temporales consecutivos. Cada cuadro (201) está dividido en 8 porciones (en este caso, porciones rectangulares) denominadas "teselas" T1 a T8. El número y la forma de las teselas pueden ser diferentes. A continuación, se considera que la teselación es la misma cualquiera que sea el índice del cuadro de vídeo.

El resultado de esta teselación son 8 sub-vídeos (202) independientes. Estos sub-vídeos representan una partición del vídeo completo. Cada sub-vídeo independiente puede ser codificado como un flujo de bits independiente, según los estándares AVC o HEVC, por ejemplo. El sub-vídeo también puede formar parte de un solo flujo de bits de vídeo, tal como, por ejemplo, teselas del estándar HEVC o cortes del estándar AVC.

El estándar HEVC define diferentes subdivisiones espaciales de imágenes: teselas, cortes y segmentos de corte. Estas diferentes subdivisiones (o particiones) han sido introducidas para diferentes propósitos: los cortes están relacionados con aspectos de transmisión en continuo, mientras que las teselas y los segmentos de corte han sido definidos para un procesamiento en paralelo.

Una tesela define una zona rectangular de una imagen que contiene un número entero de unidades de árbol de codificación (Coding Tree Units, CTU). La **figura 3** muestra la teselación de una imagen (300) definida por límites de fila y columna (301, 302). Esto hace de las teselas buenas candidatas para la descripción de zonas de interés en términos de posición y tamaño. No obstante, la organización del flujo de bits estándar de HEVC en términos de sintaxis y su encapsulación en unidades de capa abstracta de red (Network Abstract Layer, NAL) está basada más bien en cortes (como en el estándar AVC).

Según el estándar HEVC, un corte es un conjunto de segmentos de corte, donde como mínimo el primer segmento de corte es un segmento de corte independiente, y los demás, si existen, son segmentos de corte dependientes. Un segmento de corte contiene un número entero de CTU consecutivas (en el orden de escaneo de rasterizado). No tiene necesariamente una forma rectangular (por lo que es menos apropiado que las teselas para representar la zona de interés). Un segmento de corte se codifica en el flujo de bits de HEVC como una cabecera denominada "slice_segment_header" seguida de datos denominados "slice_segment_data". Los segmentos de corte independientes y los segmentos de corte dependientes se diferencian por su cabecera: los segmentos de corte dependientes tienen una cabecera más corta debido a que reutilizan información de la cabecera del segmento de corte independiente. Tanto los segmentos de corte independientes como los dependientes contienen una lista de puntos de entrada en el flujo de bits: ya sea a teselas o a puntos de sincronización de decodificación entrópica.

La figura 3 muestra diferentes configuraciones de las imágenes 310 y 320 de cortes, segmentos de corte y

teselas. Estas configuraciones difieren de la configuración de imagen 300 en la que una tesela tiene un corte (que contiene solo un segmento de corte independiente). La imagen 310 está dividida en dos teselas verticales (311, 312) y un corte (con 5 segmentos de corte). La imagen 320 está dividida en dos teselas (321, 322), la tesela izquierda 321 que tiene dos cortes (cada uno con dos segmentos de corte), la tesela derecha 322 que tiene un corte (con dos segmentos de corte). El estándar HEVC define reglas de organización entre teselas y segmentos de corte que pueden ser resumidas de la siguiente manera (se tienen que cumplir una o ambas condiciones):

- todas las CTU en un segmento de corte pertenecen a la misma tesela, y
- todas las CTU en una tesela pertenecen al mismo segmento de corte.

Para tener transporte y soporte de zona de interés coincidentes, se prefiere la configuración 300, en la que una tesela contiene un corte con un segmento independiente. No obstante, la solución de encapsulación funcionaría con las otras configuraciones 310 o 320.

Si bien la tesela es el soporte apropiado para las zonas de interés, el segmento de corte es la entidad que realmente se colocará en las unidades de NAL para el transporte en la red y se añadirá para formar una unidad de acceso (imagen codificada o muestra a nivel de formato de archivo). Según el estándar HEVC, el tipo de unidad de NAL se especifica en una cabecera de unidad de NAL. Para unidades de NAL del tipo "segmento de corte codificado", el slice_segment_header indica por medio del elemento de sintaxis "slice_segment_address" la dirección del primer bloque del árbol de codificación en el segmento de corte. La información de teselación se proporciona en una unidad de NAL de PPS (conjunto de parámetros de imagen, Picture Parameter Set). La relación entre un segmento de corte y una tesela se puede deducir a partir de estos parámetros.

Por definición, en las fronteras de las teselas, las predicciones espaciales se restablecen. Sin embargo, nada impide que una tesela utilice predictores temporales de una tesela diferente en el o los cuadros de referencia. Para construir teselas independientes, en el momento de la codificación, los vectores de movimiento para las unidades de predicción dentro de una tesela están obligados a permanecer en la tesela ubicada conjuntamente en el o los cuadros de referencia. Además, los filtros en bucle (desbloqueo y SAO) tienen que ser desactivados en las fronteras de las teselas para que no se introduzca ninguna deriva del error al decodificar solo una tesela. Este control de los filtros en bucle ya está disponible en el estándar HEVC y se establece en cabeceras de segmentos de corte con el indicador denominado "loop_filter_across_tiles_enabled_flag". Estableciendo explícitamente este indicador en 0, los píxeles en las fronteras de las teselas no dependen de los píxeles que caen en la frontera de las teselas vecinas. Cuando se cumplen las dos condiciones sobre vectores de movimiento y sobre filtros en bucle, se dice que las teselas son "decodificables de manera independiente" o "independientes".

Cuando una secuencia de vídeo se codifica como un conjunto de teselas independientes, puede ser decodificada utilizando una decodificación basada en teselas de un cuadro a otro sin correr el riesgo de perder datos de referencia o de propagar errores de reconstrucción. Esta configuración posibilita reconstruir solo una parte espacial del vídeo original que corresponde, por ejemplo, a una zona de interés.

A continuación, se consideran teselas independientes.

Haciendo referencia a la **figura 4**, se describe la encapsulación de teselas en formato de archivo ISOBMFF. Por ejemplo, cada tesela es encapsulada en una pista dedicada. La información de configuración e inicialización común a todas las teselas se encapsula en una pista específica, denominada por ejemplo "pista base de teselas". El vídeo completo se encapsula, por lo tanto, como una composición de todas estas pistas, a saber, la pista base de teselas y el conjunto de pistas de tesela.

La figura 4 muestra una encapsulación a modo de ejemplo. Un modo de encapsular vídeo teselado según el estándar ISOBMFF es dividir cada tesela en una pista dedicada, encapsular la información de configuración e inicialización común a todas las teselas en una pista específica, denominada por ejemplo "pista base de teselas" y encapsular el vídeo completo como una composición de todas estas pistas: pista base de teselas más un conjunto de pistas de tesela. La encapsulación se denomina, por lo tanto, "encapsulación de tesela multipista". En la figura 4 se proporciona un ejemplo de encapsulación de tesela multipista.

La caja 401 representa la caja ISOBMFF principal "moov", y contiene la lista completa de pistas con sus identificadores. Por ejemplo, las cajas 411 a 414 representan pistas de tesela (cuatro teselas en el presente ejemplo) y la caja 420 representa la pista base de teselas. Se pueden utilizar pistas adicionales, tales como pistas de audio o de texto, y encapsularlas en el mismo archivo. Sin embargo, por razones de concisión, dichas pistas adicionales no se analizan en el presente documento.

Tal como se representa en la figura 4, los datos de tesela son divididos en pistas independientes y direccionables, de modo que cualquier combinación de pistas de tesela puede ser reconstruida fácilmente a

partir de la pista base de teselas haciendo referencia a las pistas de tesela para decodificación y visualización. La pista base de teselas también se puede denominar "pista compuesta" o "pista de referencia", puesto que está diseñada para permitir la combinación de cualesquiera teselas: una, muchas o todas las teselas. La pista base de teselas 420 contiene información común a todas las pistas de tesela y una lista de muestras 450 (solo se representa la primera en la figura 4) en una caja "mdat". Cada muestra 450 de la pista base de teselas 420 se construye por referencia a cada pista de tesela mediante el uso de extractores (451 a 454, cada uno de los cuales representa un extractor para cada tesela). Cada pista de tesela 411 a 414 representa una parte espacial del vídeo íntegro, o con todos los cuadros. La descripción de la tesela (posición, tamaño, ancho de banda, etc.) se almacena en las cajas de cabecera de pista (no representadas) de cada pista de tesela 411 a 414. La pista base de teselas y cada pista de tesela están referenciadas entre sí (405) mediante una caja "TrackReferenceBox" en cada pista. Cada pista de tesela 411 a 414 se refiere a la pista base de teselas 420 como la pista "tbas" ("tbas" es un código específico que indica una dependencia de codificación de cada pista de tesela con la pista base de teselas, en concreto dónde encontrar el parámetro "HEVCDecoderConfigurationRecord" que permite configurar el decodificador de vídeo que procesará el flujo elemental resultante del análisis sintáctico del formato de archivo). Por el contrario, para permitir la reconstrucción completa del vídeo, la pista base de teselas 420 indica una dependencia de tipo "scal" a cada pista de tesela (405). Esto es para indicar la dependencia de codificación y reflejar la definición de muestra 450 de la pista base de teselas como extractores de los datos de pistas de tesela. Estos extractores son extractores específicos que, en el momento del análisis sintáctico, pueden soportar la ausencia de datos. En la figura 4, para proporcionar una versión del archivo que pueda ser transmitida en continuo, cada pista se descompone en segmentos multimedia (431 a 434 para las pistas de tesela y 460 para la pista base de teselas). Cada segmento multimedia comprende uno o varios fragmentos de película, indicados por la caja "moof" más los datos. Para las pistas de tesela, la parte de datos corresponde a una subparte espacial del vídeo, mientras que para la pista base de teselas contiene los conjuntos de parámetros, los mensajes SEI cuando están presentes y la lista de extractores. La caja "moov" 401 en el caso de una aplicación de transmisión en continuo encajaría en un segmento de inicialización. La figura 4 muestra solo un segmento, pero las pistas se pueden descomponer en cualquier número de segmentos, siendo la limitación el que los segmentos para las pistas de tesela y para la pista base de teselas sigan la misma descomposición temporal (es decir, estén alineados temporalmente); esto es para hacer posible conmutar entre un vídeo completo y una tesela o un conjunto de teselas. La granularidad de esta descomposición temporal no se describe en el presente documento, por razones de concisión.

El formato de archivo tiene metadatos descriptivos (tales como "VisualSampleGroupEntries", por ejemplo, o tipos de referencia de pista en cajas "tref") que describen las relaciones entre las pistas, de modo que los datos correspondientes a una tesela, una combinación de teselas o todas las teselas se puedan identificar fácilmente mediante el análisis sintáctico de metadatos descriptivos.

A continuación, las imágenes fijas se describen al mismo nivel. De este modo, cuando el usuario selecciona cualquier tesela, combinación de teselas o todas las teselas de una imagen, se facilita la identificación y la extracción. En caso de que las imágenes se mezclen con datos de vídeo, la descripción se realiza en paralelo a los metadatos descriptivos del vídeo. Por lo tanto, para el mismo conjunto de datos, se proporciona una capa de indexación adicional para las imágenes (además de las capas de indexación para el vídeo y para el audio).

En los formatos de archivo de imagen fija que utilizan cajas "meta", las imágenes con la información relacionada se describen como elementos de información. Tal como se muestra en la **figura 5**, los elementos de información se enumeran en una sub-caja dedicada "ItemInfoBox" 500 de la caja "meta". Esta sub-caja proporciona el número de elementos de información presentes en el archivo. La sub-caja también proporciona para cada elemento metadatos descriptivos representados como "ItemInfoEntry" 501. Existen varias versiones 502 (0, 1, 2) de esta caja según la evolución del estándar ISO BMFF.

Los elementos "meta" no pueden ser almacenados de manera contigua en un archivo. Asimismo, no hay ninguna limitación concreta en relación con el intercalado de los datos de los elementos. Por lo tanto, dos elementos de un mismo archivo pueden compartir uno o varios bloques de datos. Esto es especialmente útil para teselas de HEVC (las teselas pueden ser almacenadas de manera contigua o no), puesto que puede hacer que sea sencillo tener un elemento por tesela decodificable de manera independiente. Este elemento indica el desplazamiento de datos en la imagen de HEVC principal y la longitud del uno o varios cortes utilizados para la tesela por medio de un ItemLocationBox.

Según las realizaciones, se puede añadir un nuevo tipo de elemento para describir una imagen de tesela, denominado por ejemplo: "hvct" o "tesela" o reutilizado de ISO/IEC 14496-15: 'hvt1'. Cada elemento que representa la imagen de tesela (cualquiera que sea el código de cuatro caracteres elegido) puede tener una referencia de tipo "tbas" al elemento 'hvc1' del que se extrae. Cada elemento tiene un identificador "item ID" 503 y se describe además en una caja "ItemlocationBox" en términos de posición de byte y tamaño en la caja de datos multimedia que contiene los datos comprimidos para las imágenes.

Dicha sintaxis hace posible que un lector de formato de archivo (o "analizador sintáctico") determine, por medio de la lista de elementos de información, cuántos elementos de información están disponibles con información sobre su tipo 504, por ejemplo "tesela" para indicar que un elemento de información es una imagen de tesela de una imagen completa.

De este modo, se hace posible seleccionar un subconjunto de elementos de información en el archivo, una combinación de los mismos, o el conjunto completo de elementos de información, para descargar solo una tesela de la imagen y la configuración de decodificador asociada, mientras se omiten las otras teselas.

En los casos en los que una tesela de HEVC depende de otra tesela de HEVC para la decodificación, la dependencia se indicará mediante una referencia de elemento de tipo "dpnd" (o cualquier código de cuatro caracteres específico que indique dependencias de codificación) tal como se describe en el documento w14123, WD de ISO/IEC 14496-15:2013 AMD 1, "Enhanced carriage of HEVC and support of MVC with depth information", MPEG 107 San José, enero de 2014.

Este documento define herramientas para asociar NALU de tesela de HEVC con descripciones de grupos de muestras que indican la posición espacial de la tesela (utilizando el descriptor "TileRegionGroupEntry"). No obstante, no existe un equivalente directo de agrupamiento de muestra para elementos de información de metadatos que pudiera permitir la reutilización de estos descriptores.

Por lo tanto, según las realizaciones, se define un elemento de descripción de tesela por cada tesela, y la tesela se vincula a su descripción utilizando una versión modificada de la caja "ItemReferenceBox", tal como se explica a continuación.

Según otras realizaciones, solo se proporciona una descripción de la teselación, preferentemente de manera genérica. De esta forma, la lista de elementos no resulta demasiado larga.

El diseño puede ser como sigue:

- permitir que algunos elementos describan un conjunto de metadatos, similares a los grupos de muestras pero específicos para cada tipo de elemento,
- para cualquier elemento, añadir la capacidad de describir un parámetro para un tipo determinado de referencia de elemento. El parámetro se interpretaría entonces en función del tipo del elemento referido (similar al tipo de agrupamiento).

Una actualización de los metadatos descriptivos para un elemento de información puede ser necesaria tal como se explica a continuación haciendo referencia a la **figura 6**.

Según el estándar ISOBMFF, el mecanismo de agrupamiento de muestras se basa en dos cajas principales que tienen un parámetro "grouping_type" como sigue:

- la caja "SampleGroupDescriptionBox" tiene un parámetro "sgpd" que define una lista de propiedades (una lista "SampleGroupEntry",
- la caja "SampleToGroupBox" tiene un parámetro "sbgp" que define una lista de grupos de muestras con su mapeo a una propiedad.

El parámetro "grouping_type" vincula una lista de grupos de muestras a una lista de propiedades, estando especificado en la lista en la caja "SampleToGroupBox" el mapeo de un grupo de muestras a una propiedad.

Para proporcionar la misma funcionalidad para los elementos de información, se tienen que describir una lista de grupos de elementos de información y una lista de propiedades. Asimismo, debería ser posible asignar cada grupo de elementos de información a una propiedad.

A continuación, se describe cómo hacer posible que dichos metadatos descriptivos se incorporen en el formato de archivo de imagen fija. En otras palabras, cómo vincular un descriptor a un elemento de imagen. Incluso si los casos de uso se describen para el formato de archivo de imagen fija de HEVC, las siguientes características se pueden utilizar en otros estándares, tales como ISO/IEC 14496-12, para asociar cualquier tipo de elemento de información con metadatos descriptivos adicionales.

Según las realizaciones, la caja "ItemInformationEntry" 601 con el parámetro "infe" se amplía con un nuevo número de versión (602 y 603), con el fin de vincular cada elemento a una propiedad mediante un nuevo parámetro denominado "iref_type" 604, tal como se muestra en la figura 6. Esto permite evitar la creación de nuevas cajas y mejora la descripción, al mismo tiempo que la mantiene breve.

La definición original de la caja ItemInformationEntry viene dada por:

```
if (version == 2) {
 unsigned int(16) item_ID;
 unsigned int(16) item_protection_index;
 unsigned int(32) item_type;
 string item_name ;
 if (item_type=='mime') {
  string content_type;
  string content_encoding; //optional
 } else if (item_type == 'uri ') {
  string item_uri_type; }
  }
```

Una nueva versión que vincula una imagen de tesela a su descripción puede ser como sigue:

```
if ((version == 2) || (version == 3)) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index;
unsigned int(32) item_type;
string item_name;
if (version == 2) {
 if (item_type=='mime') {
  string content_type;
  string content_encoding; //optional
 } else if (item_type == 'uri ') {
  string item_uri_type; }
}
if (version == 3) {
 unsigned int(32) item_iref_parameter_count;
 for (i=0 ; i< item_iref_parameter_count ; i++) {
  unsigned int(32) iref_type;
  unsigned int(32) iref_parameter; }
}
}
```

Según otras realizaciones, más cerca de la caja "SampleToGroupBox", la definición de la caja "ItemInformationBox" con código de cuatro caracteres "iinf" se cambia como sigue, por ejemplo introduciendo una nueva versión de esta caja:

la versión actual:

```
aligned(8) class ItemInfoBox extends FullBox('iinf', version = 0, 0) {
 unsigned int(16) entry_count;
 ItemInfoEntry[entry_count ] item_infos;
 }
```

se cambia a:

```
aligned(8) class ItemInfoBox extends FullBox('iinf', version = 1, 0) {
 unsigned int(16)group_entry_count;
 for (int g=0; g< group_entry_count;g++){
   unsigned int(16) item_run;
   unsigned int(16) grouping_type;
   unsigned int(16) property_index;
   unsigned int(16) entry_count;
   ItemInfoEntry[ entry_count ] item_infos; }
 unsigned int(16) remaining_entry_count;
 ItemInfoEntry[remaining_entry_count ] item_infos;
 }
```

Alternativamente, para indicar si el grupo está en uso o no, la versión actual se cambia a:

```
aligned(8) class ItemInfoBox extends FullBox('iinf', version = 1, 0) {
  unsigned int(1)group_is_used;
  if (group_is_used == 0){ // standard iinf box but with 1 additional byte
overhead
```

```
    unsigned int(7)reserved; // for byte alignment
    unsigned int(16) entry_count;
    ItemInfoEntry[ entry_count ] item_infos;
  } else {
    unsigned int(15)group_entry_count;
    for (int g=0; g< group_entry_count;g++){
      unsigned int(16) item_run;
      unsigned int(16) grouping_type;
      unsigned int(16) property_index;
      unsigned int(16) entry_count;
      ItemInfoEntry[ entry_count ] item_infos;
    }
    unsigned int(16) remaining_entry_count;
    ItemInfoEntry[remaining_entry_count ] item_infos;
  }
}
```

El parámetro "group_entry_count" define la cantidad de grupos de elementos de información en el archivo multimedia. Para cada grupo de elementos de información, se indica una cantidad de elementos de información, empezando desde item_ID=0. Puesto que los elementos de información no tienen relaciones ni limitaciones de tiempo, a diferencia de las muestras, el módulo de encapsulación puede asignar los identificadores de los elementos de información en cualquier orden. Asignando números de identificadores crecientes después del grupo de elementos, la lista de grupos de información se puede representar de manera más eficiente utilizando un parámetro item_run, que identifica las series de identificadores de elementos de información consecutivos en un grupo.

Los elementos de información relacionados tienen un índice denominado, por ejemplo, "property_index". Este parámetro "property_index" asociado con el parámetro "grouping_type" permite a un analizador sintáctico de formato de archivo (o "lector") identificar una referencia a metadatos descriptivos o los propios metadatos descriptivos. La **figura 7** muestra dos realizaciones a modo de ejemplo.

La característica de grupo en la caja "SingleItemTypeReferenceBox" 701 se puede utilizar con una identificación de grupo "group_ID", en lugar de la identificación de elemento de información (item_ID) que se utiliza habitualmente para el valor del parámetro from_item_ID. Por diseño, la caja "SingleItemTypeReferenceBox" facilita la búsqueda de todas las referencias de un tipo específico o desde un elemento específico. Su uso con un "group_ID" en lugar de un "item_ID" permite encontrar un grupo de elementos para identificar fácilmente todas las referencias de un tipo específico. Ventajosamente, puesto que hay como máximo una caja "ItemInformationBox" por cada archivo encapsulado, no hay necesidad de definir identificaciones de grupo. El módulo de encapsulación (durante la codificación) y el módulo de análisis sintáctico (durante la decodificación) pueden ejecutar un contador respectivo (como la variable "g" en la caja "ItemInformationBox") en la lista de grupos de elementos de información a medida que se crean o leen. Alternativamente, se puede informar al analizador sintáctico, utilizando el indicador "group_used_flag", de si mantener o no el contador de identificación de grupo.

Volviendo al ejemplo con un grupo de elementos de información correspondientes a las imágenes de tesela, un grupo puede contener cuatro entradas, y la referencia 700 "SingleItemTypeReference" puede indicar la lista de elementos de información 704 de los que dependen los cuatro elementos de información de imagen de tesela y, por lo tanto, para un tipo de referencia 703 concreto.

Según otras realizaciones a modo de ejemplo, el elemento de información se utiliza en un nuevo tipo de caja "ItemReferenceBox", tal como se describe a continuación, que hace posible, a partir de un elemento 722, enumerar múltiples tipos de referencia 723 para diversos elementos de información 724 adicionales.

Para este último caso, la caja "ItemReferenceBox" 721 específica se puede implementar como sigue:

```
aligned(8) class MultipleItemTypeReferenceBox(void) extends
Box(void) {
  unsigned int(16) from_item_ID;
  unsigned int(16) reference_count;
  for (j=0; j<reference_count; j++) {
    unsigned int(32) reference_type; // new parameter to allow
multiple types
    unsigned int(16) to_item_ID;
  }
}
```

En cuanto a la caja estándar "ItemInformationBox", se describe la lista de entradas de elementos, pero esta vez con un orden diferente según el agrupamiento. En el ejemplo de tesela, esto puede conducir a un primer grupo de cuatro elementos de información correspondientes a las imágenes de tesela reunidas en un grupo con un parámetro que se puede denominar "tesela", seguido de elementos de información no agrupados para la información de configuración, para el elemento de información de imagen completa y, opcionalmente, para los metadatos EXIF.

De este modo, se modifica una caja y se crea una caja que es un tipo específico de ItemReferenceBox. A continuación, se describe este nuevo tipo de ItemReferenceBox.

La caja "ItemReferenceBox" también se puede ampliar distinguiendo entre los distintos tipos de ItemReferenceBox mediante el uso de los parámetros de indicador en la caja "FullBox" que forma parte de "ItemReferenceBox" de la siguiente manera:

```
aligned(8) class ItemReferenceBox extends FullBox('iref', 0, flags) {
switch (flags) {
case 0:
 SingleItemTypeReferenceBox references[];
 break;
 case 1:
 MultipleItemTypeReferenceBox references[];
 break;
 case 2:
 SharedItemTypeReferenceBox references[];
 break;
 }
 }
```

Usando la caja "MultipleItemTypeReferenceBox" 721, una imagen con cuatro teselas se puede describir como sigue:

```
Item Reference Box (version=1 or flags=1):
fromID=2, ref_count=1, type='cdsc', toID=1;
fromID=1, ref_count=1, type='init', toID=3;
fromID=4, ref_count=2, type='tbas', toID=1, type='tile' toID=8;
fromID=5, ref_count=2, type='tbas', toID=1, type='tile' toID=8;
fromID=6, ref_count=2, type='tbas', toID=1, type='tile' toID=8;
fromID=7, ref_count=2, type='tbas', toID=1, type='tile' toID=8;
```

Este diseño hace que sea bastante más fácil encontrar todas las referencias de cualquier tipo de un elemento específico.

El soporte de descripción 711 para una lista de elementos 712 que hacen referencia a un mismo elemento 714 con un tipo 713 determinado puede ser como sigue:

```
aligned(8) class SharedItemTypeReferenceBox(ref_type) extends
Box(referenceType) {
 unsigned int(16) reference_count;
 for (j=0; j<reference_count; j++) {
   unsigned int(16) from_item_ID; }
 unsigned int(16) to_item_ID;
 }
 }
```

En el ejemplo de una imagen con cuatro teselas, por lo tanto, se puede tener:

```
type='cdsc', ref_count=1, fromID=2, toID=1;
 type='init', ref_count=1, fromID=1, toID=3;
 type='tbas', ref_count=4, fromID=4, fromID=5, fromID=6, fromID=7,
 toID=1;
 type='tile', ref_count=4, fromID=4, fromID=5, fromID=6, fromID=7,
 toID=8;
```

El diseño de la caja "SharedItemTypeReferenceBox" facilita la búsqueda de todas las referencias de un tipo

específico que apuntan a un elemento específico. Esto contrasta con la caja "SingleItemTypeReferenceBox". Pero, puesto que la mayoría de las "reference_type" definidas para referencias de pista no son bidireccionales, la caja "SingleItemTypeReferenceBox" no se puede utilizar con algún tipo de referencia unidireccional para señalizar todos los elementos que tienen este tipo de referencia a otros elementos. Alternativamente, se puede proporcionar un indicador en la "SingleItemTypeReference" para indicar si es una referencia directa o una referencia inversa, aliviando de este modo la necesidad de la nueva SharedItemTypeReferenceBox.

En vista de lo anterior, un elemento de información se puede asociar con información de teselación. A continuación, se tiene que proporcionar una descripción de esta información de teselación.

Por ejemplo, cada tesela se puede describir utilizando un descriptor de tesela, tal como el "iref_parameter" 605 de la "ItemInfoEntry" 601 extendida. Un descriptor específico puede ser como sigue:

```
aligned(8) class TileInfoDataBlock() {
unsigned int(8) version;
unsigned int(32) reference_width; // full image sizes
unsigned int(32) reference_height;
unsigned int(32) horizontal_offset; // tile positions
unsigned int(32) vertical_offset;
unsigned int(32) region_width; // tile sizes
unsigned int(32) region_height;
}
```

Según las realizaciones, se puede utilizar un descriptor para aplicar la cuadrícula de teselas a las una o varias imágenes a almacenar.

Dicho descriptor puede ser como sigue:

```
aligned(8) class TileInfoDataItem () {
unsigned int(8) version;
unsigned int(1) regular_spacing; // regular grid or not
unsigned int(7) reserved = 0;
unsigned int(32) reference_width; // full-frame sizes
unsigned int(32) reference_height;
unsigned int(32) nb_cell_horiz;
unsigned int(32) nb_cell_vert;
if (!regular_spacing) {
 for (i=0; i<nb_cell_width; i++)
  unsigned int(16) cell_width;
 for (i=0; i<nb_cell_height; i++)
  unsigned int(16) cell_height; }
}
}
```

Este descriptor "TileInfoDataItem" permite describir una cuadrícula de teselación (regular o irregular). La cuadrícula se describe fila por fila comenzando desde la parte superior izquierda.

El descriptor se almacenará como un elemento de tipo "tesela". Cuando otro elemento hace referencia a este elemento, utilizará una referencia de tipo "tesela" a esta descripción, y tendrá un parámetro "iref_parameter" especificado, cuyo valor es el índice basado en 0 de la celda en la cuadrícula definida por el descriptor, donde 0 es el elemento superior izquierdo, 1 es la celda inmediatamente a la derecha de la celda 0, y así sucesivamente.

En el descriptor:

- "version" (versión) indica la versión de la sintaxis para el TileInfoDataItem. Solo se define el valor 0.
- "regular_spacing" indica si todas las teselas de la cuadrícula tienen la misma anchura y la misma altura.
- "reference_width, reference_height" indica las unidades en las que se describe la cuadrícula. Estas unidades pueden coincidir o no con la resolución de píxeles de la imagen a la que se refiere este elemento. Si la cuadrícula es regular, "reference_width" (resp. "reference_height") debe ser un múltiplo de "nb_cell_horiz" (resp. "nb_cell_vert").
- "cell_width" proporciona la división horizontal de la cuadrícula en teselas no regulares, comenzando desde la izquierda.
- "cell_height" proporciona la división vertical de la cuadrícula en teselas no regulares, comenzando desde arriba.

El enfoque anterior permite compartir la información de teselación para todas las teselas.

Además, en caso de que haya múltiples imágenes que compartan la misma teselación, se puede compartir incluso más descripción simplemente haciendo referencia a una celda en la cuadrícula de teselas.

La configuración de teselación se puede colocar en la caja de datos multimedia o en una caja dedicada compartida (por referencia) entre los elementos de información de tesela.

Los descriptores anteriores son descriptores espaciales puros en el sentido de que solo proporcionan ubicaciones espaciales y tamaños para una o varias subimágenes en una imagen más grande. En algunos casos de uso, por ejemplo, con conjuntos de imágenes o composición de imágenes, una ubicación espacial no es suficiente para describir la imagen, habitualmente cuando las imágenes se superponen. Esta es una limitación del descriptor TileInfoDataBlock anterior. Para permitir la composición de la imagen, sea cual sea la imagen, es decir, una tesela o una imagen independiente/completa, puede ser útil definir un descriptor que contenga, por un lado, las posiciones y tamaños de la imagen (relaciones espaciales) y, por otro lado, información de visualización (color, recorte...) para esa imagen. Por ejemplo, se puede proporcionar información de color para transformar una subimagen de un espacio de color a otro, para visualización. Este tipo de información se puede transmitir en la ColorInformationBox "colr" del ISOBMFF. Puede ser útil, por conveniencia, tener los mismos datos preparados para diferentes tipos de visualización simplemente proporcionando los parámetros de transformación a aplicar en lugar de transmitir las dos imágenes diferentes transformadas de este modo. Asimismo, la relación de aspecto de píxel como la caja PixelAspectRatio "pasp" definida en la Parte 12 del ISOBMFF se puede colocar en este descriptor para redefinir una anchura y una altura que pueden ser diferentes de la anchura y la altura codificadas de cada imagen. Esto indicaría la relación de escala que se aplicará en la pantalla después de la decodificación de una imagen. Por lo tanto, se tendrían los tamaños codificados almacenados en las entradas de muestra de vídeo (caja "stsd" por ejemplo) y los tamaños de pantalla deducidos de la caja "pasp". Otra información posible para la visualización podría ser la caja de información de apertura limpia "clap" también definida en ISOBMFF. Según el estándar SMPTE 274M, la apertura limpia define un área dentro de la cual la información de la imagen está subjetivamente no contaminada por todas las distorsiones transitorias de borde (posibles efectos de zumbido en las fronteras de las imágenes después de las conversiones de analógica a digital). Esta lista de parámetros útiles para la visualización no es limitativa y se podría poner como componentes opcionales en el descriptor de subimagen cualquier otra caja de metadatos descriptivos. Estos pueden ser mencionados explícitamente porque ya forman parte del estándar y proporcionan herramientas genéricas para indicar el recorte de la imagen, la modificación de la relación de aspecto de la muestra y los ajustes de color. Desgraciadamente, su uso solo era posible para pistas multimedia, no para formatos de archivo de imagen que dependen de cajas "meta". Por lo tanto, se sugiere un nuevo descriptor denominado, por ejemplo, "SimpleImageMetaData" para soportar la descripción espacial de los elementos de imagen, junto con otras propiedades, tales como la apertura limpia o la relación de aspecto de la muestra. Esto aplica a cualquier subimagen (tesela o imagen independiente) destinada a ser compuesta en una imagen más grande o, al revés, extraída de una imagen más grande:

```
aligned(8) class SimpleImageMetaData {
 CleanApertureBox clap; // optional
 PixelAspectRatioBox pasp; // optional
 ColourInformationBox colour; // optional
 ImageSpatialRelationBox location; // optional
 }
```

O su variación cuando se consideran los parámetros de extensión para ayudar al proceso de visualización (a través, por ejemplo, de extra_boxes):

```
aligned(8) class SimpleImageMetaData {
 CleanApertureBox        clap; // optional
 PixelAspectRatioBox     pasp; // optional
 ColourInformationBox    colour; // optional
 ImageSpatialRelationBox location; // optional
 extra_boxes             boxes; // optional
}
```

Donde ImageSpatialRelationBox es una extensión de TileinfoDataBlock tal como se describe a continuación. Otro parámetro útil a considerar es la posibilidad de componer imágenes como capas. A continuación, se sugiere insertar un parámetro para indicar el nivel asociado a una imagen en esta composición en capas. Esto suele ser útil cuando las imágenes se superponen. Esto se puede denominar "capa" para un ejemplo con indicación de información de capa. Se proporciona un ejemplo de sintaxis para dicho descriptor:

Definición:

Tipo de caja: “isre”
Contenedor: elemento de metadatos de imagen individual (“simd”)
Mandatorio: no
Cantidad: cero o uno por cada elemento

Sintaxis:

```
aligned(8) class ImageSpatialRelationBox
extends FullBox('isre, version = 0, 0) {
 unsigned int(32) horizontal_display_offset;
 unsigned int(32) vertical_display_offset;
 unsigned int(32) display_width;
 unsigned int(32) display_height;
 int(16) layer;
 }
```

con la semántica asociada:

Horizontal_display_offset especifica el desplazamiento horizontal de la imagen.
Vertical_display_offset especifica el desplazamiento vertical de la imagen.
Display_width especifica la anchura de la imagen.
Display_height especifica la altura de la imagen.
Layer especifica el orden de adelante atrás de la imagen; las imágenes con números más bajos están más cerca del espectador. 0 es el valor normal y -1 estaría delante de la capa 0, y así sucesivamente.

Este nuevo tipo de caja “isre” da la capacidad de describir la posición relativa de una imagen con otras imágenes en un conjunto de imágenes. Proporciona un subconjunto de las funcionalidades de la matriz de transformación que normalmente se encuentra en la caja de cabecera de película o pista de un archivo multimedia. Las coordenadas en ImageSpatialRelationBox se expresan en una cuadrícula que proporciona el tamaño de visualización previsto por el autor del conjunto; estas unidades pueden coincidir o no con el tamaño codificado de la imagen. El tamaño de visualización previsto está definido por:

- Horizontalmente: el valor máximo de (horizontal_display_offset + display_width) para todas las cajas “isre”
- Verticalmente: el valor máximo de (vertical_display_offset + display_height) para todas las cajas “isre”

Cuando algunas imágenes no tienen ningún “isre” asociado mientras que otras imágenes en el archivo sí lo tienen, las imágenes predeterminadas sin ningún “isre” se tratarán como si sus desplazamientos horizontales y verticales fueran 0, su tamaño de visualización fuera el tamaño de visualización previsto y su capa fuera 0.

El ImageSpatialRelationBox indica la posición espacial relativa de las imágenes después de que se haya aplicado a las imágenes cualquier recorte o relación de aspecto de muestra. Esto significa que, cuando “isre” se combina con “pasp”, etc. en un SimpleImageMetaData, la imagen se decodifica, se aplican “pasp”, “clap”, “colr” si están presentes y, a continuación, la imagen se desplaza y escala al desplazamiento y tamaño declarados en la caja “isre”.

Este nuevo descriptor se puede utilizar como descripción de una imagen (tesela o imagen individual) definiendo una asociación entre la información del elemento que representa la imagen y la información del elemento que representa el descriptor (asígnese el tipo “simd” para la definición de SimpleImageMetadata, cualquier código reservado de 4 caracteres sería aceptable para un analizador sintáctico de mp4 para identificar fácilmente el tipo de metadatos que está procesando actualmente). Esta asociación se realiza con un ItemRefererenceBox y con un nuevo tipo de referencia; “simr” para indicar “relación de imagen espacial”. La descripción de ejemplo a continuación muestra el caso de una composición de 4 imágenes donde la composición en sí no tiene ningún elemento asociado. Cada elemento de imagen está asociado a un elemento SimpleImageMetaData por medio de una referencia de elemento de tipo “simr”, y comparte la información de DecoderConfigurationRecord en un elemento “hvcC” dedicado.

```
ftyp box: major-brand = 'heve', compatible-brands = 'heve'
meta box: (container)
  handler box: hdlr = 'hvci' // no primary item provided
  Item Information Entries:
  item_type = 'hvc1', itemID=1, item protection index = 0
  item_type = 'hvc1', itemID=2, item protection index = 0
  item_type = 'hvc1', itemID=3, item protection index = 0
  item_type = 'hvc1', itemID=4, item protection index = 0
  item_type='simd' itemID=5 (sub-image descriptor)
```

```
item_type='simd' itemID=6 (sub-image descriptor)
item_type='simd' itemID=7 (sub-image descriptor)
item_type='simd' itemID=8 (sub-image descriptor)
item type = 'hvcC', item ID=9, item protection _index = 0...
Item Reference:
type='simr' fromID=1, toID=5
type='simr' fromID=2, toID=6
type='simr' fromID=3, toID=7
type='simr' fromID=4, toID=8
type='init', fromID=1, toID=9;
type='init', fromID=3, toID=9;
type='init', fromID=4, toID=9;
type='init', fromID=5, toID=9;
   Item Location:
   itemID = 1, extent count = 1, extent offset = P1, extent length = L1;
   itemID = 2, extent count = 1, extent offset = P2, extent length = L2;
   itemID = 3, extent count = 1, extent offset = P3, extent length = L3;
   itemID = 4, extent count = 1, extent offset = P4, extent length = L4;
   itemID = 5, extent count = 1, extent offset = P5, extent length = L5;
   itemID = 6, extent count = 1, extent offset = P6, extent length = L6;
   itemID = 7, extent count = 1, extent offset = P7, extent length = L7;
   itemID = 8, extent count = 1, extent offset = P8, extent length = L8;
   itemID = 9, extent_count = 1, extent_offset = P0, extent_length = L0;
   Media data box:
1 HEVC Decoder Configuration Record ('hvcC' at offset P0)
4 HEVC Images (at file offsets P1, P2, P3, P4)
4 simple image metadata (at file offsets P5, P6, P7, P8)
```

La organización de datos anterior se proporciona como ejemplo: imagen y metadatos podrían estar entrelazados en la caja de datos multimedia, por ejemplo, para tener una imagen más sus metadatos direccionables como un solo intervalo de bytes. Al recibir esta descripción, se informa a un analizador sintáctico, mediante el análisis sintáctico de la información en los elementos "simd", si una subimagen está recortada de una imagen completa o, por el contrario, si una imagen completa es una composición de subimágenes. En caso de recorte, el elemento de imagen completa y la imagen recortada compartirían el mismo intervalo de datos que en el ejemplo siguiente, y la misma información de configuración del decodificador. La subimagen se asociaría por lo tanto a un elemento "simd" que solo tiene información "clap" y no tiene posicionamiento, por lo tanto no tiene "isre".

En caso de composición: en dicho caso, el elemento de imagen completa se asocia a un elemento "simd" que solo contiene información "isre", y la subimagen se asociaría a un elemento "simd" que refleja su posición en la imagen completa.

El ejemplo siguiente muestra el caso en el que 4 imágenes se componen en una más grande. Todas las imágenes, incluida la compuesta, se exponen como un elemento reproducible utilizando el descriptor propuesto.

```
ftyp box: major-brand = 'heve', compatible-brands = 'heve'
meta box: (container)
  handler box: hdlr = 'hvcl' primary item: itemID = 1;
  Item Information Entries:
  item_type = 'hvcl', itemID=1, item protection_index = 0... // full-image
  item_type = 'hvcl', itemID=2, item_protection_index = 0... // sub-image
  item_type = 'hvcl', itemID=3, item_protection_index = 0... // sub-image
  item_type = 'hvcl', itemID=4, item_protection_index = 0... // sub-image
  item_type = 'hvcl', itemID=5, item_protection_index = 0... // sub-image
  item type = 'simd' itemID=6 (sub-image descriptor)...
  item_type = 'simd' itemID=7 (sub-image descriptor)...
  item_type = 'simd' itemID=8 (sub-image descriptor)...
  item type = 'simd' itemID=9 (sub-image descriptor)...
  item_type = 'hvcC', item_ID=10 (decoder config record)
  item_type = 'simd', item ID=11 (sub-image descriptor)
  Item Reference Entries:
  type= 'simr', fromID=1, toID=11
```

```
type= 'simr', fromID=1, toID=11
type= 'simr', fromID=2, toID=6
type= 'simr', fromID=3, toID=7
type= 'simr', fromID=4, toID=8
type= 'simr', fromID=5, toID=9
type= 'init', fromID=1, toID=10...
type= 'init', fromID=2, toID=10...
type= 'init', fromID=3, toID=10...
type= 'init', fromID=4, toID=10...
type= 'init', fromID=5, toID=10...
   Item Location:
   itemID = 1, extent count = 4,11 full image is composed of 4 sub-images
   extent_offset = P2, extent length = L2;
   extent offset = P3, extent length = L3;
   extent offset = P4, extent length = L4;
   extent offset = P5, extent length = L5;
   itemID = 2, extent count = 1, extent_offset = P2, extent length = L2;
   itemID = 3, extent count = 1, extent offset = P3, extent length = L3;
   itemID = 4, extent count = 1, extent offset = P4, extent length = L4;
   itemID = 5, extent count = 1, extent offset = P5, extent length = L5;
   itemID = 6, extent_count = 1, extent_offset = P6, extent_length = L6;
   itemID = 7, extent count = 1, extent offset = P7, extent length = L7;
   itemID = 8, extent count = 1, extent offset = P8, extent length = L8;
   itemID = 9, extent count = 1, extent offset = P9, extent length = L9;
   itemID = 10, extent_count = 1, extent_offset = P0, extent_length = L0;
   itemID = 11, extent _count = 1, extent _offset = P10, extent_length =
   L10;
   Media data box:
1 HEVC Decoder Configuration Record ('hvcC' at offset P0)
4 HEVC (sub) Images (at file offsets P2, P3, P4, P5)
5 simple image metadata (at file offsets P6, P7, P8, P9, P10)
```

Este otro ejemplo muestra el caso en el que la imagen completa es en realidad una imagen de HEVC teselada (4 teselas):

```
ftyp box: major-brand = 'heve', compatible-brands = 'heve'
meta box: (container)
handler box: hdlr = 'hvc1' primary item: itemID = 1;
Item Information Entries:
item_type = 'hvc1', itemID=1, item_protection_index = 0... // full-image
item_type = 'hvt1', itemID=2, item_protection_index = 0... // sub-image
item_type = 'hvt1', itemID=3, item_protection_index = 0... // sub-image
item_type = 'hvt1', itemID=4, item_protection_index = 0... // sub-image
item_type = 'hvt1', itemID=5, item protection index = 0... // sub-image
item type = 'simd' itemID=6 (sub-image descriptor)...
item_type = 'simd' itemID=7 (sub-image descriptor)...
item type = 'simd' itemID=8 (sub-image descriptor)...
item_type = 'simd' itemID=9 (sub-image descriptor)...
item_type = 'hvcC', item ID=10 (decoder config record)
Item Reference Entries:
type= 'init', fromID=1, toID=10...
// declare sub-images as tiles of the full image
type= 'tbas', fromID=2, toID=1...
type= 'tbas', fromID=3, toID=1...
type= 'tbas', fromID=4, toID=1...
type= 'tbas', fromID=5, toID=1...
// providing positions and sizes
type= 'simr', fromID=2, toID=6
type= 'simr', fromID=3, toID=7
type= 'simr', fromID=4, toID=8
type= 'simr', fromID=5, toID=9
   Item Location:
   itemID = 1, extent count = 4,11 full image is composed of 4 tiles
   extent_offset = P2, extent _length = L2... // data for tile 1
```

```
    extent_offset = P3, extent_length = L3... // data for tile 2
    extent_offset = P4, extent_length = L4... // data for tile 3
    extent_offset = P5, extent_length = L5... // data for tile 4
    itemID = 2, extent_count = 1, extent_offset = P2, extent_length = L2;
    itemID = 3, extent_count = 1, extent_offset = P3, extent_length = L3;
    itemID = 4, extent_count = 1, extent_offset = P4, extent_length = L4;
    itemID = 5, extent_count = 1, extent_offset = P5, extent_length = L5;
    itemID = 6, extent_count = 1, extent_offset = P6, extent_length = L6;
    itemID = 7, extent_count = 1, extent_offset = P7, extent_length = L7;
    itemID = 8, extent_count = 1, extent_offset = P8, extent_length = L8;
    itemID = 9, extent_count = 1, extent_offset = P9, extent_length = L9;
    itemID = 10, extent_count = 1, extent_offset = P0, extent_length = L0;
    Media data box:
1 HEVC Decoder Configuration Record ('hvcC' at offset P0)
1 HEVC Image (with 4 tiles at file offsets P2, P3, P4, P5)
4 simple image metadata (at file offsets P6, P7, P8, P9)
```

Dependiendo de los casos de uso, sería posible tener varios elementos de imagen que compartan los mismos metadatos, por ejemplo, cuando se tiene que aplicar el mismo recorte a todas las imágenes. También es posible que un elemento de imagen tenga múltiples referencias "simr" a diferentes SimpleImageMetaData, por ejemplo, cuando el recorte se comparte entre imágenes, pero no la información espacial.

Una realización alternativa a la nueva versión de ItemInfoEntry (tal como se muestra en la figura 6) es definir más de un parámetro (605) por cada entrada y referencia de elemento de información. En la realización de la figura 6, el iref_parameter es un código de cuatro bytes que es útil en el caso de un índice de tesela para hacer referencia a una celda en una cuadrícula de teselación. Pero para tener una descripción más rica y poder incrustar una descripción vinculada dentro de la entrada de información del propio elemento, en lugar de con los datos (en la caja mdat), la siguiente extensión puede ser útil:

```
if (version == 3) {
unsigned int(32) item iref parameter count;
for (i=0 ; i< item iref _parameter_count ; i++) {
 unsigned int(32) iref_type;
 ItemReferenceParameterEntry parameter;
 }
 aligned(8) abstract class ItemReferenceParameterEntry (unsigned int(32)
 format)
extends Box(format) {
}
// Example to reference a tile index
aligned(8) abstract class TileIndexItemReferenceParameterEntry
extends ItemReferenceParameterEntry('tile') {
unsigned int(32) tile_index; }
// Example to inline the tile description
aligned(8) abstract class TileIndexItemReferenceParameterEntry
extends ItemReferenceParameterEntry('tile') {
unsigned int(32) tile_index; }
```

En la extensión anterior:

- item_iref_parameter_count proporciona la cantidad de tipos de referencia para los que se proporciona un parámetro. Esto no cambia en comparación con el elemento 605 en la figura 6.
-
- iref_type proporciona el tipo de referencia, tal como se indica en la caja "iref", para el que se aplica el parámetro para este elemento. Esto no ha cambiado en comparación con el elemento 605 de la figura 6.
-
- el parámetro en este caso difiere de iref_parameter (elemento 605 de la figura 6) debido a que proporciona un medio de extensión por medio de la nueva caja ItemReferenceParameterEntry. Especializando esta nueva caja (tal como se hizo anteriormente con TileIndexItemReferenceParameterEntry para el índice de tesela en una configuración de teselación), cualquier tipo de metadatos adicionales se pueden asociar con una entrada de elemento de información siempre que los módulos de encapsulación y análisis sintáctico conozcan la estructura de esta caja especializada. Esto se puede realizar mediante tipos estándar de ItemReferenceParameterEntry o proporcionando por construcción, o en una etapa de negociación, la estructura de la entrada de parámetro. La semántica del parámetro viene dada por la semántica del elemento con tipo iref_type.

A continuación, se dan a conocer metadatos descriptivos, a modo de ejemplo, para elementos de información que describen una imagen con 4 teselas y los metadatos EXIF de la imagen completa.

En la técnica anterior, las imágenes de tesela se enumeraban como elementos de información sin proporcionar ninguna descripción correspondiente, tal como se muestra a continuación. Además, la información de configuración designada como tipo "hvcC" no se describía como un elemento. Esto hace posible factorizar los datos comunes relacionados con los conjuntos de parámetros de HEVC y los mensajes de SEI que se aplican a todas las imágenes de tesela y a la imagen completa.

```
ftyp box: major-brand = 'heve', compatible-brands = 'heve'
meta box: (container)
  handler box: hdlr = 'hvcl' primary item: itemID = 1;
  Item information:
  item_type = 'hvcl', itemID=1, item_protection_index = 0 (unused) =>
  Full pict.
  item_type = 'Exif', itemID=2, item_protection_index = 0 (unused)
  item_type = 'hvcC', itemID=3, item_protection_index = 0 (unused)
  item_type = 'hvct', itemID=4, item_protection_index = 0 (unused) =>
  Tile pict.
  item_type = 'hvct', itemID=5, item_protection_index = 0 (unused) =>
  Tile pict.
  item_type = 'hvct', itemID=6, item_protection_index = 0 (unused) =>
  Tile pict.
  item_type = 'hvct', itemID=7, item_protection_index = 0 (unused) =>
  Tile pict.
  Item Location:
  itemID = 1, extent count = 1, extent offset = X, extent length = Y;
  itemID = 2, extent count = 1, extent offset = P, extent length = Q;
  itemID = 3, extent count = 1, extent offset = R, extent length = S;
  itemID = 4, extent count = 1, extent offset = X, extent length = ET1;
  itemID = 5, extent _count = 1, extent _offset = X+ET1, extent_length =
  ET2;
  itemID = 6, extent_count = 1, extent_offset = X+ET2, extent_length =
  ET3;
  itemID = 7, extent_count = 1, extent_offset = X+ET3, extent_length =
  ET4;
Item Reference:
type='cdsc', fromID=2, toID=1;
type='init', fromID=1, toID=3;
type='tbas', fromID=4, toID=1;
type='tbas', fromID=5, toID=1;
type='tbas', fromID=6, toID=1;
type='tbas', fromID=7, toID=1;
Media data box:
HEVC Image (at file offset X, with length Y)
Exif data block (at file offset P, with length Q)
HEVC Config Record (at file offset R, with length S)
// No Tile description
```

Según las realizaciones, utilizando la extensión con la versión 3 (véase la figura 6, 602, 603) de la caja ItemInfoEntry (601): se enumera información de imagen de tesela con referencias asociadas a partes de la configuración de teselación que también se describe como un elemento de información (ID=8).

```
ftyp box:   major-brand = 'hevc', compatible-brands = 'hevc'
meta box:   (container)
   handler box:   hdlr = 'hvcl'        primary item: itemID = 1;
   Item information:
item_type = 'hvcl', itemID=1, item_protection_index = 0 (unused)
item_type = 'Exif', itemID=2, item_protection_index = 0 (unused)
item_type = 'hvcC', itemID=3, item_protection_index = 0 (unused)
item_type = 'hvct', itemID=4, parameter for ireftype==tile:
tile_index=0
item_type = 'hvct', itemID=5, parameter for ireftype==tile:
```

```
tile_index=1
item_type = 'hvct', itemID=6, parameter for ireftype==tile:
tile_index=2
item_type = 'hvct', itemID=7, parameter for ireftype==tile:
tile_index=3
item_type = 'tile', itemID=8, (tiling configuration)

   Item Location:
itemID = 1, extent_count = 1, extent_offset = X, extent_length = Y;
itemID = 2, extent_count = 1, extent_offset = P, extent_length = Q;
itemID = 3, extent_count = 1, extent_offset = R, extent_length = S;
itemID = 4, extent_count = 1, extent_offset = X, extent_length = ET1;
itemID = 5, extent_count = 1, extent_offset = X+ET1, extent_length =
ET2;
itemID = 6, extent_count = 1, extent_offset = X+ET2, extent_length =
ET3;
itemID = 7, extent_count = 1, extent_offset = X+ET3, extent_length =
ET4;
itemID = 8, extent_count = 1, extent_offset = i, extent_length = I;

   Item Reference:
type='cdsc', fromID=2, toID=1;
type='init', fromID=1, toID=3;
type='tbas', fromID=4, toID=1;
type='tbas', fromID=5, toID=1;
type='tbas', fromID=6, toID=1;
type='tbas', fromID=7, toID=1;
type='tile', fromID=4, toID=8; //
type='tile', fromID=5, toID=8; // link each tile pict.
type='tile', fromID=6, toID=8; // to the tiling config item
type='tile', fromID=7, toID=8; //

Media data box:
   HEVC Image (at file offset X, with length Y)
   Exif data block (at file offset P, with length Q)
   HEVC Config Record (at file offset R, with length S)
   Tile description data block (at file offset i, with length I)
```

La **figura 8** muestra un contexto de implementación de realizaciones de la invención. En primer lugar, se registran diferentes medios: por ejemplo, audio durante la etapa 800a, vídeo durante la etapa 800b, y una o varias imágenes durante la etapa 800c. Cada medio se comprime durante las respectivas etapas 801a, 801b y 801c. Durante estas etapas de compresión se generan los flujos elementales 802a, 802b y 802c. A continuación, a nivel de aplicación (selección del usuario desde la interfaz gráfica de usuario; configuración del sistema de generación multimedia, etc.), se selecciona un modo de encapsulación para determinar si todos estos flujos elementales deben ser fusionados o no. Cuando se activa el modo "fusionar" (prueba 803, "sí"), los datos de audio, vídeo e imágenes fijas se encapsulan en el mismo archivo durante la etapa 806c tal como se ha descrito anteriormente. Si el modo "fusionar" no está activado (prueba 803, "no"), entonces se generan dos archivos encapsulados durante las etapas 806a y 806b consecutivamente o en paralelo, lo que conduce respectivamente a la creación de un archivo de datos multimedia en tiempo sincronizado durante la etapa 807a, y un archivo adicional con solo las imágenes fijas 907b. Durante la etapa 806a, los flujos elementales de audio y vídeo se encapsulan según el estándar ISOBMFF y las imágenes fijas se encapsulan durante la etapa 806b, tal como se ha descrito anteriormente en este documento, con el fin de proporcionar una descripción de tesela y características de la zona de interés. Finalmente, se obtiene una presentación multimedia 807, y se le puede proporcionar a un generador de DASH para prepararla para su transmisión en continuo (etapa 820a) o almacenarla en una memoria (etapa 820b) o renderizarla en una unidad de visualización (etapa 820c) o transmitirla (etapa 820d) a una entidad remota, ya sea íntegramente o después de que algunas partes (tal como teselas) hayan sido extraídas analizando sintácticamente los metadatos descriptivos.

De acuerdo con descripciones anteriores de realizaciones, cabe señalar que los metadatos descriptivos, tales como por ejemplo la caja SimpleImageMetadata ("simd") (también denominados ISOBMFFMetaData en la última versión de la especificación de formato de archivo de imagen fijas), se describen como elementos

completos. Los metadatos descriptivos o prescriptivos adicionales también se definen en la especificación de formato de archivo de imágenes fijas, tal como se describe en el documento w14878, estudio del comité de ISO/IEC 23008-12:2013 1.ª edition, "Information technology - MPEG systems technologies - Part 12: Image File Format", MPEG 110 Estrasburgo, octubre de 2014. Ejemplos de metadatos descriptivos o prescriptivos son CleanApertureBox ("clap"), ImageRotation ("irot"), ExifDataBlock ("exif") o ImageOverlay ("iovl"). En términos más generales, los metadatos descriptivos son metadatos que proporcionan información adicional o descripción para un elemento como una imagen o una subimagen (por ejemplo, metadatos Exif), y los metadatos prescriptivos son operaciones o transformaciones que se aplicarán a un elemento (por ejemplo, una rotación, un recorte o una combinación de varios elementos que forman los operadores de transformación).

Sin embargo, puede resultar bastante molesto tener que almacenar dichos metadatos descriptivos o prescriptivos en la especificación como elementos completos; estos son solo pseudo-elementos, que requieren que los metadatos descriptivos o prescriptivos se almacenen con datos codificados en la caja mdat (110), y que se definan entradas en itemlocationBox (iloc) (109), itemInfoBox (iinf) e itemProtectionBox (ipro). Requerir esas entradas en iloc, iinf e ipro para esto es una sobrecarga bastante grande. Por ejemplo, una entrada en itemInfoBox requiere el uso de una caja completa como mínimo con una cabecera de 12 bytes; además, se tiene que definir un item_protection_index (16 bits) más un item_name (8 bits) vacío para un total de 15 bytes de coste adicional por cada entrada en itemInfoBox (iinf). Una entrada en itemlocationBox (iloc) también requiere como mínimo 9 bytes en los mejores casos (base_offset_size= offset_size= length_size= 1, 1 extensión). En la práctica, la entrada itemlocationBox se utiliza con base_offset_size = offset_size=length_size=2 o 4, lo que significa 12 o 18 bytes de coste adicional. Además, estos metadatos suelen ser pequeños y permiten una lectura eficiente de los demás elementos. Tenerlos almacenados como elementos dedicados puede complicar el análisis sintáctico de archivos, especialmente la obtención parcial de un archivo (multiplicación de solicitudes de HTTP, por ejemplo).

En una realización alternativa según la invención reivindicada, todos los metadatos descriptivos y prescriptivos pueden estar definidos como elementos integrados que pueden ser almacenados en la caja meta (100) como parte de otras cajas en lugar de en la caja mdat (110) y, por lo tanto, pueden evitar el coste adicional de definir las entradas itemInfoBox e itemlocationBox.

Para almacenar metadatos descriptivos y prescriptivos en la caja meta, se define una caja de elementos virtuales denominada "VirtualItemBox". Según una realización acorde con la invención reivindicada, todas las cajas de metadatos descriptivos y prescriptivos se heredan de esta clase de elemento virtual.

Un elemento virtual tiene asignados un item_ID y un item_type, junto con un conjunto de cajas. Los elementos virtuales son datos adicionales que se utilizan habitualmente para describir metadatos que se asociarán con otros elementos. Según una realización acorde con la invención reivindicada, el elemento virtual permite asociar una entrada de la itemInfoBox que identifica un elemento (imagen o subimagen) y la operación o la transformación que se aplicará a este elemento. Habitualmente, esta asociación se puede describir definiendo una entrada de tipo "simr" en la itemReferenceBox desde el item_ID de la imagen hasta el item_ID de la caja de descripción de la operación o transformación de metadatos. Los elementos virtuales solo se pueden referenciar en cajas de referencia de elemento y cajas de elemento principal, y no se deben declarar ni referenciar en ninguna otra caja (por ejemplo, itemlocationBox (iloc), itemInfoBox (iinf), itemProtectionBox (ipro)). "VirtualItemBox" se define como sigue:

```
aligned(8) class VirtualItemBox(unsigned int(32) item_type)

      extends FullBox('vite', version, 0) {

if (version == 0) {
          unsigned int(16)  item_ID;
} else {
         unsigned int(32)  item_ID;
         }
         unsigned int(32)  item_type;
}
```

con la siguiente semántica para sus parámetros:

item_ID: ID (o identificador) de este elemento. Es ilegal tener entradas en iinf, iloc o ipro con el mismo valor de item_ID
item_type: es un valor de 32 bits, habitualmente 4 caracteres imprimibles, que es un indicador de tipo de

elemento válido definido, como “mime”.

Opcionalmente, en una variante, “VirtualItemBox” también puede incluir un parámetro adicional denominado “descriptor_family”. La familia de descriptores indica si la caja de metadatos son metadatos descriptivos o prescriptivos. En una variante, la familia de descriptores indica el tipo de caja de metadatos a partir de una lista de valores predefinidos. Por ejemplo: transfo_operator, composed_image, descriptive_metadata.

Heredando de esta caja de elementos virtuales, todas las cajas de metadatos descriptivos y prescriptivos se pueden almacenar en la caja meta sin la necesidad de definir entradas asociadas en itemInfoBox (iinf) e itemlocationBox (iloc) pero aún conservan la ventaja de ser direccionables por las cajas de referencia de elemento.

Según esta realización, ImageOverlay (iovl), SubSampleItemData (subs), AuxiliaryConfiguration (auxC), ExifDataBlock (exif), SimpleImageMetadata (simd) y el elemento de imagen derivada se heredan de la clase de elemento virtual.

También según una realización de la invención reivindicada, se introduce un tipo de elemento individual y genérico denominado “dimg”, con referencias de elemento de tipo “simr” a elementos de tipo “simd”. Este enfoque permite la reutilización de propiedades cuando sea apropiado y reduce el número de elementos y referencias de elementos. Se añade ImageRotationBox a los SimpleImageMetadata (simd). El tipo de referencia “simr” define un vínculo desde un elemento de imagen hacia un elemento “simd”, para proporcionar acceso directo a los metadatos descriptivos de la imagen.

Además, la caja de metadatos de ImageOverlay (iovl) se rediseña de la siguiente manera para que ya no dependa del orden de referencia.

```
aligned(8) class ImageOverlay {
   unsigned int(8) version = 0;
   unsigned int(8) flags;
   for (j=0; j<3; j++) {
         unsigned int(16) canvas_fill_value;
   }

FieldLength = ((flags & 1) + 1) * 16;
unsigned int(FieldLength) output_width;
unsigned int(FieldLength) output_height;
for (i=0; i<reference_count; i++) {
      unsigned int(16) item_id;
      signed int(FieldLength) horizontal_offset;
      signed int(FieldLength) vertical_offset;
}
}
```

Se añade un item_id explícito para cada entrada en el bucle para identificar explícitamente el elemento que está compuesto.

En una realización alternativa, todas las cajas incluidas en SimpleImageMetadata (simd) se definen como cajas de metadatos independientes que se heredan de la caja de elementos virtuales.

En una realización alternativa, la rotación de imagen individual se puede declarar integrando la operación de rotación directamente en la caja del descriptor de metadatos de imagen SimpleImageMetadata (“simd”) (también denominada ISOBMFFMetaData en la última versión de la especificación de formato de archivo de imagen fija) como sigue:

```
aligned(8) class ISOBMFFMetaData {
   CleanApertureBox  clap;                    // optional
   PixelAspectRatioBox pasp;                  // optional
   ColourInformationBox colour;               // optional
   ImageSpatialRelationBox location;          // optional
    ImageRotationBox rotation;                // optional
   Box extra_boxes[];                         // optional
}


aligned(8) class ImageRotationBox
extends FullBox('irot', version = 0, flags = 0) { // 12 extra-
bytes
   unsigned int (6) reserved = 0;
   unsigned int (2) angle;
}
```

Aunque la caja de rotación es ligeramente más grande que los elementos "irot" (12 bytes), el beneficio de usar este enfoque es claro cuando se combinan transformaciones, tales como rotación y CleanApperture, puesto que solo se necesita un "simd", en lugar de una cascada de elementos derivados.

En dicho caso, el elemento derivado genérico, "dimg" (descrito anteriormente), se puede usar para hacer referencia tanto al elemento de imagen como a la descripción de metadatos. Un elemento de este tipo podría entonces ser enumerado como elemento principal en PrimaryItemBox ("pitm").

Otro beneficio de este enfoque es que un autor puede indicar claramente que solo desea que se muestre el elemento rotado.

Los párrafos siguientes proponen una alternativa a la realización descrita anteriormente, fuera del alcance de la invención reivindicada.

Esta alternativa es ventajosamente simple en lo que respecta a cómo se pueden aplicar las transformaciones (o "efectos") a imágenes en el formato de archivo de imagen fija ISO. En concreto, se resuelven los siguientes problemas con esta realización alternativa:

- el elevado número de referencias de elementos;
- el creciente número de elementos cuando se aplican efectos en cascada; y
- la imposibilidad de mutualizar los efectos para un conjunto determinado de elementos, es decir, un conjunto de imágenes o porciones de imágenes como la zona de interés.

Las soluciones existentes propusieron mutualizar los efectos como diferentes extensiones (es decir, desplazamientos de bytes en la parte de datos) del elemento. Más en detalle, extensión significa que una imagen derivada se describiría como una lista de extensiones en la itemlocationBox ("iloc"), identificando cada extensión un fragmento de la parte de datos ("mdat"), correspondiendo cada fragmento a uno o varios metadatos descriptivos o prescriptivos o de transformación.

Pero varios inconvenientes son inherentes a esta solución:

- la creación de un archivo de imágenes encapsuladas resulta bastante complicada: tocar un efecto en un elemento de imagen derivada implica inspeccionar todas las imágenes derivadas para verificar si comparten la misma extensión y potencialmente reescribir parte de ella;
- el análisis sintáctico tampoco es muy simple, puesto que el lector del archivo de imágenes necesitará averiguar si una cadena de transformaciones/efectos es la misma en diferentes elementos en dicho archivo (sin señalización directa);
- para cada transformación/efecto se necesitará una nueva extensión en la itemlocationBox ("iloc") siempre que la nueva transformación/efecto no se almacene continuamente con la transformación/efecto en la cadena de transformaciones/efectos a aplicar. Además, la combinación o conexión en cascada de efectos

puede resultar costosa cuando no se almacenan en extensiones contiguas en la parte de datos.

Además, estas soluciones requerían almacenamiento de implementación, lo que implica la creación de una caja para almacenar el efecto, a fin de comprender su tipo (hasta ahora, el tipo de efecto lo proporcionaba el item_type). Definiendo un nuevo formato de caja para el efecto, una solución más sencilla es definir los efectos por separado de los elementos y tener un mapeo directo entre elementos y efectos sin ningún coste adicional.

La realización alternativa propone una simplificación del manejo de los efectos teniendo una separación limpia en los formatos de archivo:

- elementos regulares (imágenes o porciones de imágenes) (por ejemplo: hvc1, ...) vinculados con sus metadatos descriptivos (tal como se propuso anteriormente: ya sea por medio del tipo de referencia "init" o "simr" o cualquier tipo de referencia que describa metadatos descriptivos);
- "imágenes derivadas", que son un conjunto de efectos (o transformaciones) aplicados a uno o varios elementos fuente (imagen o porción de imagen) identificados por medio de una referencia de elemento "dimg" desde el elemento de "imagen derivada" al elemento fuente; y
- una estructura que representa las transformaciones/efectos, incluyendo un conjunto de varios efectos diferentes.

Las ventajas de esta realización alternativa son:

- la reusabilidad de efectos: declarados una vez y potencialmente referenciados varias veces
- descripciones más compactas definiendo conjuntos de efectos (más sobre esto a continuación);
- legibilidad general, incluyendo que no se necesitan nuevas extensiones de la itemlocationBox; y
- manteniendo pequeña la cantidad de referencias de elementos.

Según esta realización alternativa, se define un nuevo elemento derivado único con el tipo de elemento "dimg". Este elemento derivado único se representa concretamente por:

```
aligned(8) class DerivedImage  {
  bit(2) index_mode;
     bit (6) reserved;
  if (index_mode==0) nb_bits_effect = 8;
  else if (index_mode==1) nb_bits_effect = 16;
  else if (index_mode==2) nb_bits_effect = 32;

  unsigned int(nb_bits_effect) nb_effects;
  for (i=0; i<nb_effects; i++) {
  unsigned int(nb_bits_effect) effect_id;
}
}
```

Donde nb_effects representa el número de efectos que se aplicarán a una imagen de origen para componer la imagen derivada, y effect_id es un identificador único en el archivo encapsulado del efecto que se aplicará. Los efectos se aplican en orden inverso a su aparición en la lista de efectos.

La imagen derivada o el elemento transformado denominado "DerivedImage" define una imagen como un conjunto de efectos que se aplicarán a una imagen de origen antes de ser presentada a un usuario o a una pantalla de visualización, por ejemplo. La imagen de origen se identifica mediante una referencia de elemento de tipo "dimg" (o cualquier tipo de referencia reservado) desde el elemento derivado hasta la imagen de origen. La propia imagen de origen puede ser cualquier elemento de imagen (imágenes o porciones de imágenes, superposición de imágenes, imagen derivada) definido en la especificación de formato de archivo de imagen fija ISO. No debe haber más de una referencia de elemento "dimg" del mismo elemento (pero puede haber varias en el mismo elemento, si este elemento se reutiliza varias veces para diferentes composiciones).

La imagen derivada se almacena en la parte de datos del archivo.

Al editar el archivo encapsulado, por ejemplo, eliminando un efecto de un archivo de imagen, se deben eliminar todas las referencias a este efecto de las imágenes derivadas.

Los efectos se pueden aplicar a imágenes, porciones de imágenes, imágenes compuestas o imágenes derivadas por medio de los elementos DerivedImage. Cada efecto se describe mediante una caja que se deriva de una estructura BaseEffectBox mostrada a continuación.

```
class BaseEffetBox(effect_type) extends
FullBox(effect_type, version, flags){
  if (version==0) nb_bits_effect = 8;
  else if (version ==1) nb_bits_effect = 16;
  else if (version ==2) nb_bits_effect = 32;

unsigned int(nb_bits_effect) effect_id;
}
```

Con la siguiente semántica:

effect_type es el tipo de caja de efectos que se derivan de esta clase, un código único de cuatro caracteres que identifica el tipo de caja;

effect_id es un identificador único para un efecto o transformación determinados. Este identificador debe ser único dentro de la caja "meta".

nb_bits_effect se deriva del valor de la versión e indica la cantidad de bits utilizados para representar el effect_id

Los efectos se pueden declarar en una EffectDeclarationBox opcional, contenida en la caja "meta":

Tipo de caja: "effd"
Contenedor: meta
Obligatorio: no
Cantidad: vero o uno

```
class EffectDeclarationBox extends Box('effd'){
  //one or more effect boxes
}
```

Por ejemplo, se pueden definir los siguientes efectos (sin lista limitativa):

el Efecto de rotación: el efecto de rotación transforma la imagen de origen en sentido antihorario en unidades de 90 grados.

Tipo de caja: "erot"
Contenedor: effd
Mandatorio: No
Cantidad: cero o más

```
class RotationEffectBox extends
BaseEffectBox('erot'){
  unsigned int (6) reserved = 0;
  unsigned int (2) angle;
}
```

La semántica es:
ángulo *90: especifica el ángulo (en sentido antihorario) en unidades de grados

- Efecto de apertura limpia: el efecto de apertura limpia modifica la parte visible de la imagen de origen.

Tipo de caja: "ecla"
Contenedor: effd
Mandatorio: no
Cantidad: cero o más

```
class CleanApertureEffectBox extends
BaseEffectBox('ecla'){
        unsigned int(nb_bits_effect)
cleanApertureWidthN;
        unsigned int(nb_bits_effect)
cleanApertureWidthD;
        unsigned int(nb_bits_effect)
cleanApertureHeightN;
        unsigned int(nb_bits_effect)
cleanApertureHeightD;
        unsigned int(nb_bits_effect) horizOffN;
        unsigned int(nb_bits_effect) horizOffD;
         unsigned int(nb_bits_effect) vertOffN;
         unsigned int(nb_bits_effect) vertOffD;
      }
```

La semántica es:

nb_bits_effect se deriva de la clase padre BaseEffectBox e indica la cantidad de bits utilizados para representar los diferentes campos de CleanApertureEffectBox;
hSpacing, vSpacing: definen la anchura y la altura relativos de un píxel;
cleanApertureWidthN, cleanApertureWidthD: un número fraccionario que define la anchura de apertura limpia exacto, en píxeles contados, de la imagen;
cleanApertureHeightN, cleanApertureHeightD: un número fraccionario que define la altura de apertura limpia exacta, en píxeles contados, de la imagen;
horizOffN, horizOffD: un número fraccionario que define el desplazamiento horizontal del centro de apertura limpia menos (anchura-1)/2 (habitualmente 0);
vertOffN, vertOffD: un número fraccionario que define el desplazamiento vertical del centro de apertura limpia menos (altura-1)/2 (habitualmente 0).

El conjunto de efectos: la caja Conjunto de efectos permite definir un conjunto de varios efectos como un solo efecto, con el fin de reutilizarlo para varias imágenes y reducir de este modo el coste de la descripción en términos de bytes.

Tipo de caja: "ecol"
Contenedor: effd
Mandatorio: No
Cantidad: cero o más

```
class EffectCollectionBox extends
BaseEffectBox('ecol') {
    unsigned int(nb_bits_effect) nb_effects;
    for (i=0; i<nb_effects; i++) {
    unsigned int(nb_bits_effect) apply_effect_id;
  }
  }
```

La semántica es:

nb_bits_effect se deriva de la clase padre BaseEffectBox, e indica la cantidad de bits utilizados para representar los diferentes campos de EffectCollectionBox.
apply_effect-id: indica el ID de un efecto que se aplicará a la imagen de origen.

Los efectos en un conjunto de efectos se aplican en el mismo orden que los efectos en el elemento DerivedImage; por ejemplo, cada efecto se aplicará a la entrada en el orden inverso de su aparición en la lista de efectos.

La OverlayEffectBox declara una composición de imágenes como una superposición. Para este efecto específico, la imagen derivada resultante no tiene referencia a ninguna imagen de origen, puesto que este efecto declara la lista de imágenes de origen que forman parte de la composición.

```
class OverlayEffectBox extends BaseEffectBox('eovl') {
    bit(1) fill_required;
    bit(7) reserved;
     if (fill_required) {
          for (j=0; j<3; j++) {
               unsigned int(nb_bits_effects)
               canvas_fill_value;
          }
     }
    unsigned int(nb_bits_effects) output_width;
    unsigned int(nb_bits_effects) output_height;
    unsigned int(nb_bits_effects) nb_images;
     for (i=0; i<nb_images; i++) {
          unsigned int (nb_bits_effects) image_item_ID;
          signed int(nb_bits_effects) horizontal_offset;
          } } signed int(nb_bits_effects) vertical_offset;
```

con la siguiente semántica:

nb_bits_effects se deriva de la clase padre BaseEffectBox e indica la cantidad de bits utilizados para representar los diferentes campos de OverlayEffectBox;
fill_required indica si hay huecos en la imagen compuesta resultante para rellenar con un valor de fondo;

canvas_fill_value: indica el valor de píxel por cada canal utilizado si ningún píxel de ninguna imagen de entrada se encuentra en una ubicación de píxel concreta. Si las imágenes de entrada contienen menos de tres canales, la semántica de canvas_fill_value correspondiente a los canales que no están presentes en las imágenes de entrada no se especifica;
nb_images indica la cantidad de imágenes a componer, cada una identificada por su item_ID, tal como indica el parámetro image_item_ID.
output_width, output_height: especifican la anchura y la altura, respectivamente, de la imagen de salida en la que se colocan las imágenes de entrada. El área de imagen de la imagen de salida se conoce como el lienzo.

horizontal_offset, vertical_offset: especifica el desplazamiento, desde la esquina superior izquierda del lienzo, hasta el cual se ubica la imagen de entrada. Las ubicaciones de píxeles con un valor de desplazamiento negativo no se incluyen en la imagen de salida. Las ubicaciones de píxeles horizontales mayores o iguales que output_width no se incluyen en la imagen de salida. Las ubicaciones de píxeles verticales mayores o iguales que output_height no se incluyen en la imagen de salida.

La **figura 9** es un diagrama de bloques esquemático de un dispositivo informático 900 para la implementación de una o varias realizaciones de la invención. El dispositivo informático 900 puede ser un dispositivo tal como un micro-ordenador, una estación de trabajo o un dispositivo portátil ligero. El dispositivo informático 900 comprende un bus de comunicación conectado a:

- una unidad central de procesamiento 901, tal como un microprocesador, denominada CPU;
- una memoria de acceso aleatorio 902, denominada RAM, para almacenar el código ejecutable del procedimiento de las realizaciones de la invención, así como los registros adaptados para registrar las variables y parámetros necesarios para implementar el procedimiento para leer y escribir los manifiestos y/o para codificar el vídeo y/o para leer o generar los datos bajo un formato de archivo determinado, pudiendo ampliarse la capacidad de memoria de la misma mediante una RAM opcional conectada a un puerto de expansión, por ejemplo;
- una memoria de solo lectura 903, denominada ROM, para almacenar programas informáticos para implementar las realizaciones de la invención;
- una interfaz de red 904 está conectada habitualmente a una red de comunicación a través de la cual se transmiten o reciben datos digitales a procesar. La interfaz de red 904 puede ser una sola interfaz de red, o estar compuesta por un conjunto de diferentes interfaces de red (por ejemplo, interfaces cableadas e inalámbricas, o diferentes tipos de interfaces cableadas o inalámbricas). Los datos se escriben en la interfaz de red para transmisión o son leídos desde la interfaz de red para la recepción bajo el control de la aplicación de software que se ejecuta en la CPU 901;
- una interfaz de usuario 9805, para recibir entradas de un usuario o para mostrar información a un usuario;
- un disco duro 906, denominado HD
- un módulo de E/S 907 para recibir/enviar datos desde/hacia dispositivos externos, tales como una fuente o una pantalla de vídeo.

El código ejecutable puede ser almacenado en la memoria de solo lectura 903, en el disco duro 906 o en un medio digital extraíble, tal como por ejemplo un disco. Según una variante, el código ejecutable de los programas puede ser recibido por medio de una red de comunicación, a través de la interfaz de red 904, para ser almacenado en uno de los medios de almacenamiento del dispositivo de comunicación 900, tal como el disco duro 906, antes de ser ejecutado.

La unidad central de procesamiento 901 está adaptada para controlar y dirigir la ejecución de las instrucciones o porciones del código de software del programa o programas, según las realizaciones de la invención, instrucciones que se almacenan en uno de los medios de almacenamiento mencionados anteriormente. Tras ser encendida, la CPU 901 es capaz de ejecutar instrucciones desde la memoria RAM principal 902 relacionadas con una aplicación de software después de que esas instrucciones hayan sido cargadas desde la ROM de programa 903 o el disco duro (HD, Hard-Disc) 906, por ejemplo. Dicha aplicación de software, cuando es ejecutada por la CPU 901, hace que se realicen las etapas de un procedimiento según las realizaciones.

Alternativamente, la presente invención puede ser implementada en hardware (por ejemplo, en forma de un circuito integrado de aplicación específica o ASIC (Application Specific Integrated Circuit)).

La presente invención puede ser integrada en un dispositivo tal como una cámara, un teléfono inteligente o una tableta que actúa como un controlador remoto para un televisor, por ejemplo, para hacer zum en una zona de interés concreta. También puede ser utilizada desde los mismos dispositivos para tener una experiencia de navegación personalizada del programa de TV, seleccionando áreas de interés específicas. Otro uso desde estos dispositivos por un usuario es compartir con otros dispositivos conectados algunas subpartes seleccionadas de sus vídeos preferidos. También puede ser utilizada en un teléfono inteligente o tableta para monitorizar lo que sucedió en un área específica de un edificio sometido a vigilancia, siempre que la cámara de vigilancia admita la parte de generación de esta invención.

Si bien la invención ha sido mostrada y descrita en detalle en los dibujos y en la descripción anterior, dicha ilustración y descripción deben ser consideradas ilustrativas o a modo de ejemplo, y no limitativas, y la invención no está limitada a la realización dada a conocer. Los expertos en la materia pueden comprender y llevar a cabo otras variaciones de la realización dada a conocer poniendo en práctica la invención reivindicada, a partir del estudio de los dibujos, de la descripción y de las reivindicaciones adjuntas.

En las reivindicaciones, la expresión "que comprende" no excluye otros elementos o etapas, y los artículos indefinidos "un" o "una" no excluyen una pluralidad. Un solo procesador u otra unidad puede cumplir las funciones de varios elementos enumerados en las reivindicaciones. El mero hecho de que se citen diferentes características en reivindicaciones dependientes diferentes entre sí no indica que no se pueda utilizar ventajosamente una combinación de estas características. Cualesquiera signos de referencia en las reivindicaciones no deben ser interpretados como limitativos del alcance de la invención.

REIVINDICACIONES

1. Procedimiento de encapsulación de un flujo de bits codificado que representa una o varias imágenes, en un archivo de datos encapsulado, comprendiendo el archivo de datos encapsulado una parte de datos y una parte de metadatos, según define el estándar de encapsulación de formato de archivo multimedia basado en ISO, ISOBMFF, comprendiendo el procedimiento:

- obtener información de elemento de imagen que identifica una porción de la parte de datos, representando dicha porción una imagen individual o una subimagen de una imagen individual del flujo de bits codificado;
- obtener propiedades de imagen que comprenden una propiedad relacionada con la anchura y la altura de una o varias imágenes; y
- encapsular dicho flujo de bits codificado junto con dicha información de elemento de imagen proporcionada y las propiedades de imagen en el archivo de datos encapsulado,

en el que

- la parte de metadatos se incluye en una caja "meta" de ISOBMFF,
- las propiedades de imagen se almacenan en cajas en la parte de metadatos, estando asociada, cada una, con un identificador, y
- el identificador asociado con cada una de las cajas se utiliza para asociar entradas de una 'itemInfoBox' de ISOBMFF que identifica elementos de imagen o elementos de subimagen con las propiedades de imagen.

2. Procedimiento, según la reivindicación 1, en el que las propiedades de imagen comprenden además uno o varios parámetros de entre:

- posición de la imagen,
- relación de aspecto de píxeles,
- información de color, y
- propiedades transformativas, que comprenden una o varias propiedades de entre:

- recorte,
- rotación.

3. Procedimiento, según la reivindicación 1, en el que las entradas comprenden:

- información de tipo, y
- un identificador utilizado para vincular una información de elemento de imagen a las propiedades de imagen.

4. Procedimiento, según cualquiera de las reivindicaciones 1 a 3, en el que el identificador en las entradas de una "itemInfoBox" de ISOBMFF se utiliza para asociar elementos de imagen o elementos de subimagen con las propiedades de imagen por medio de una caja de referencia.

5. Procedimiento de procesamiento de un archivo de datos encapsulado, que comprende una parte de datos y una parte de metadatos, según se define en el estándar de encapsulación de formato de archivo multimedia basado en ISO, ISOBMFF, comprendiendo el procedimiento:

- obtener el archivo de datos encapsulado que incluye un flujo de bits codificado correspondiente a una o varias imágenes en la parte de datos y, en la parte de metadatos, información de elemento de imagen que identifica una porción de la parte de datos, representando dicha porción una imagen individual o una subimagen de una imagen individual del flujo de bits codificado, e información que incluye propiedades de imagen que comprenden una propiedad relacionada con la anchura y la altura de una o varias imágenes o subimágenes; y
- generar la una o varias imágenes o subimágenes,

en el que

- la parte de metadatos se incluye en una caja "meta" de ISOBMFF,
- las propiedades de imagen se almacenan en cajas en la parte de metadatos, estando, cada una, asociada con un identificador, y
- el identificador asociado con cada una de las cajas se utiliza para asociar entradas de una "itemInfoBox" de ISOBMFF que identifica elementos de imagen o elementos de subimagen con propiedades de imagen.

6. Procedimiento, según la reivindicación 5, en el que un identificador en las entradas de una "itemInfoBox" de ISOBMFF se utiliza para asociar elementos de imagen o elementos de subimagen con propiedades de imagen por medio de una caja de referencia

7. Dispositivo para encapsular un flujo de bits codificado que representa una o varias imágenes, que comprende una memoria configurada para almacenar los datos de imagen y un dispositivo configurado para implementar un procedimiento según cualquiera de las reivindicaciones 1 a 4.

8. Dispositivo para procesar un flujo de bits codificado encapsulado que representa una o varias imágenes, que comprende una memoria configurada para almacenar el flujo de bits codificado encapsulado y un dispositivo configurado para implementar un procedimiento según la reivindicación 5 o 6.

9. Programa informático que comprende instrucciones que, cuando son ejecutadas en un ordenador, hacen que dicho ordenador realice el procedimiento según cualquiera de las reivindicaciones 1 a 6.

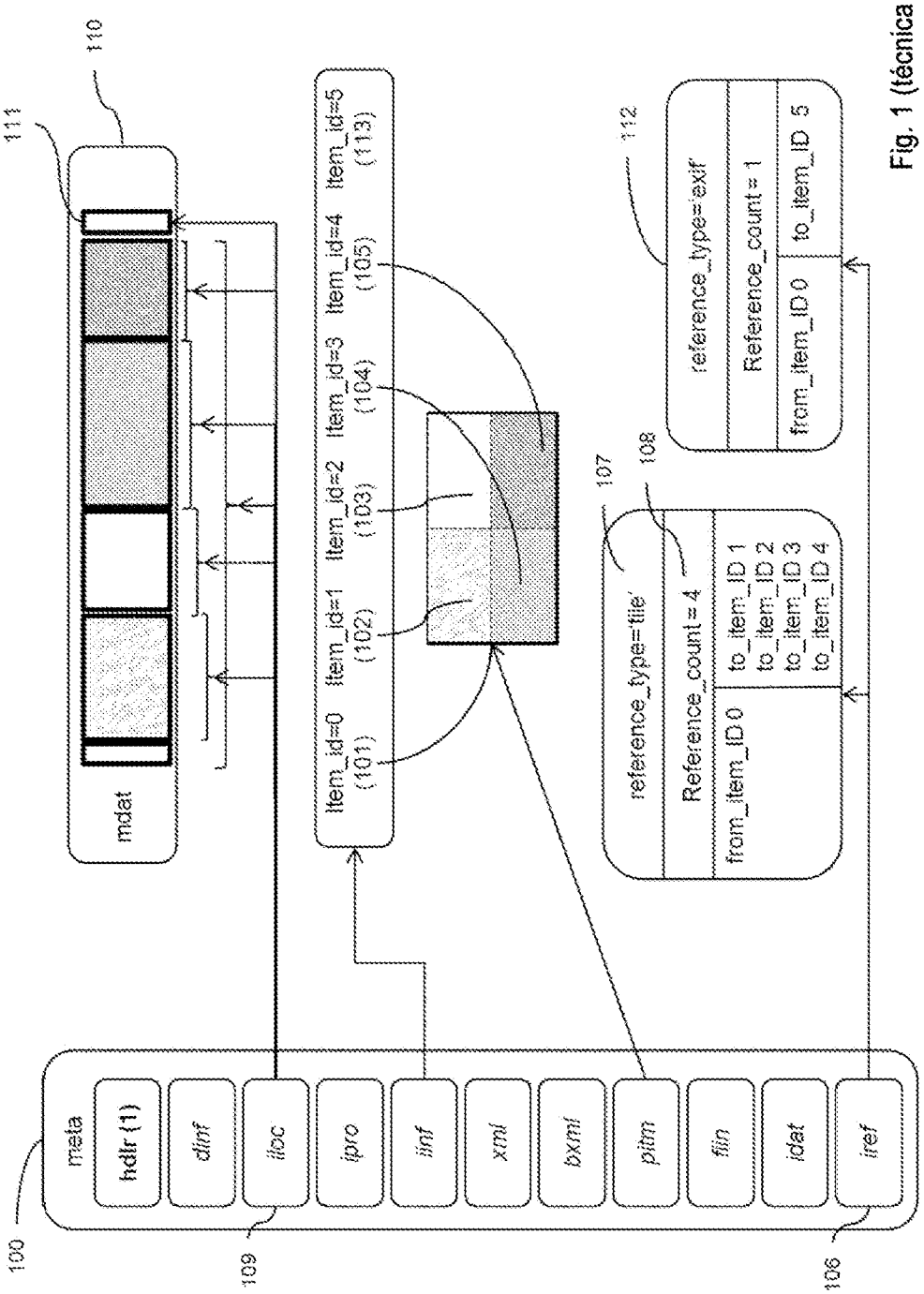

Fig. 1 (técnica anterior)

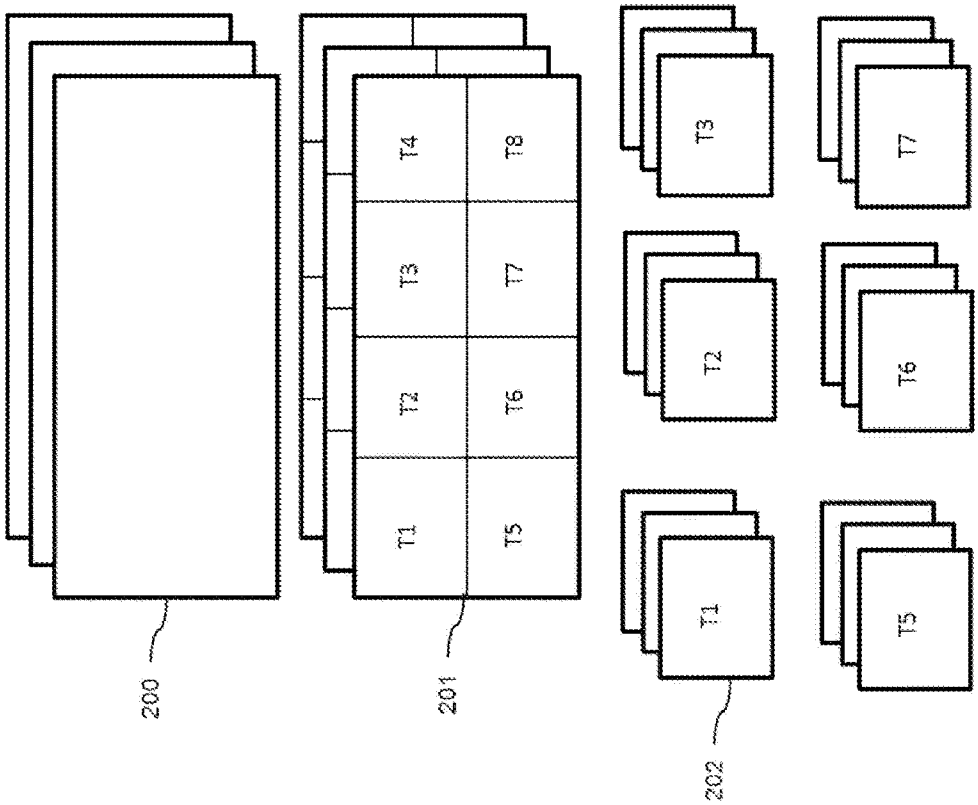

Fig. 2

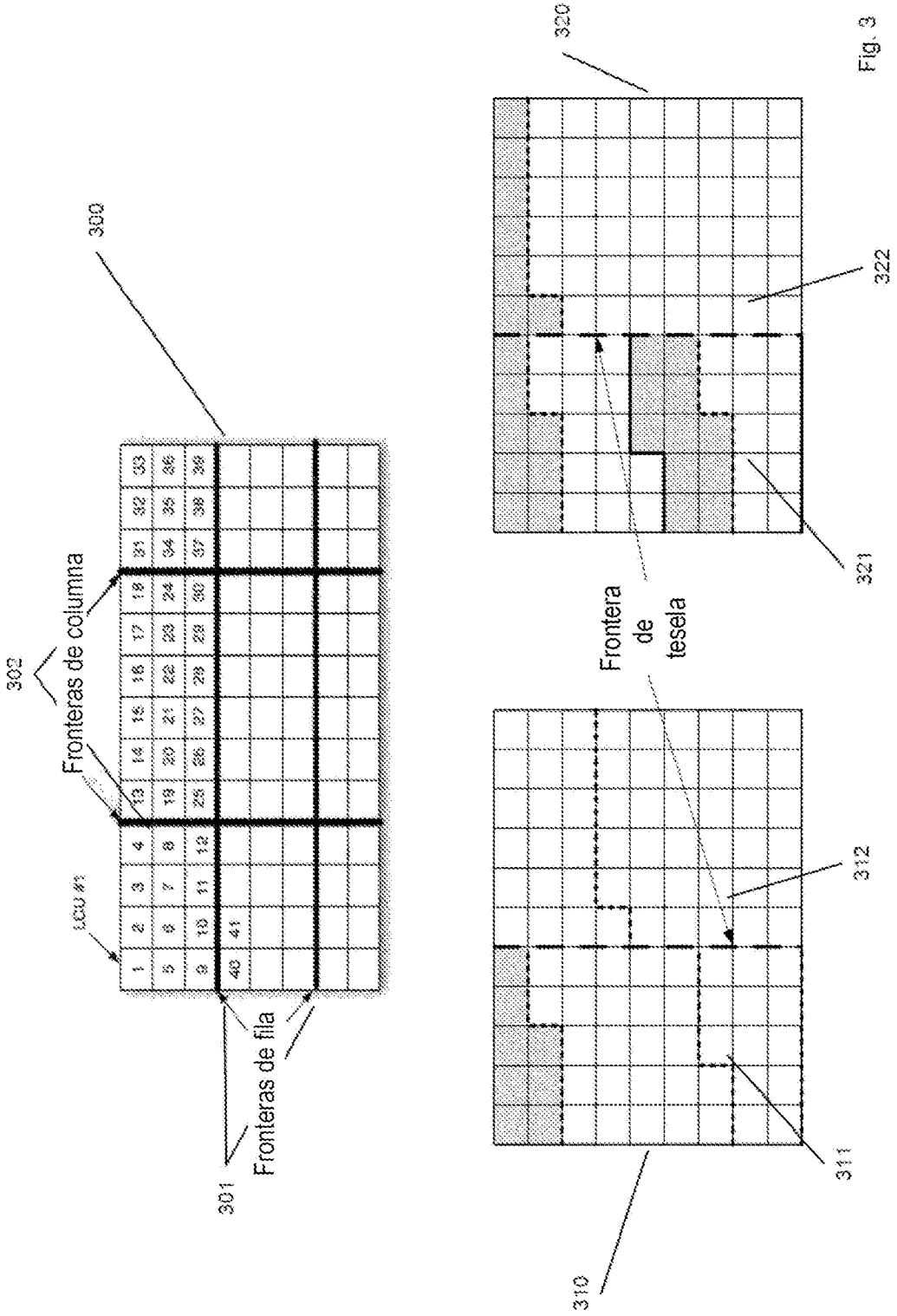
300
302
Fronteras de columna
LCU #1
301
Fronteras de fila
320
322
321
Frontera de tesela
312
311
310

Fig. 3

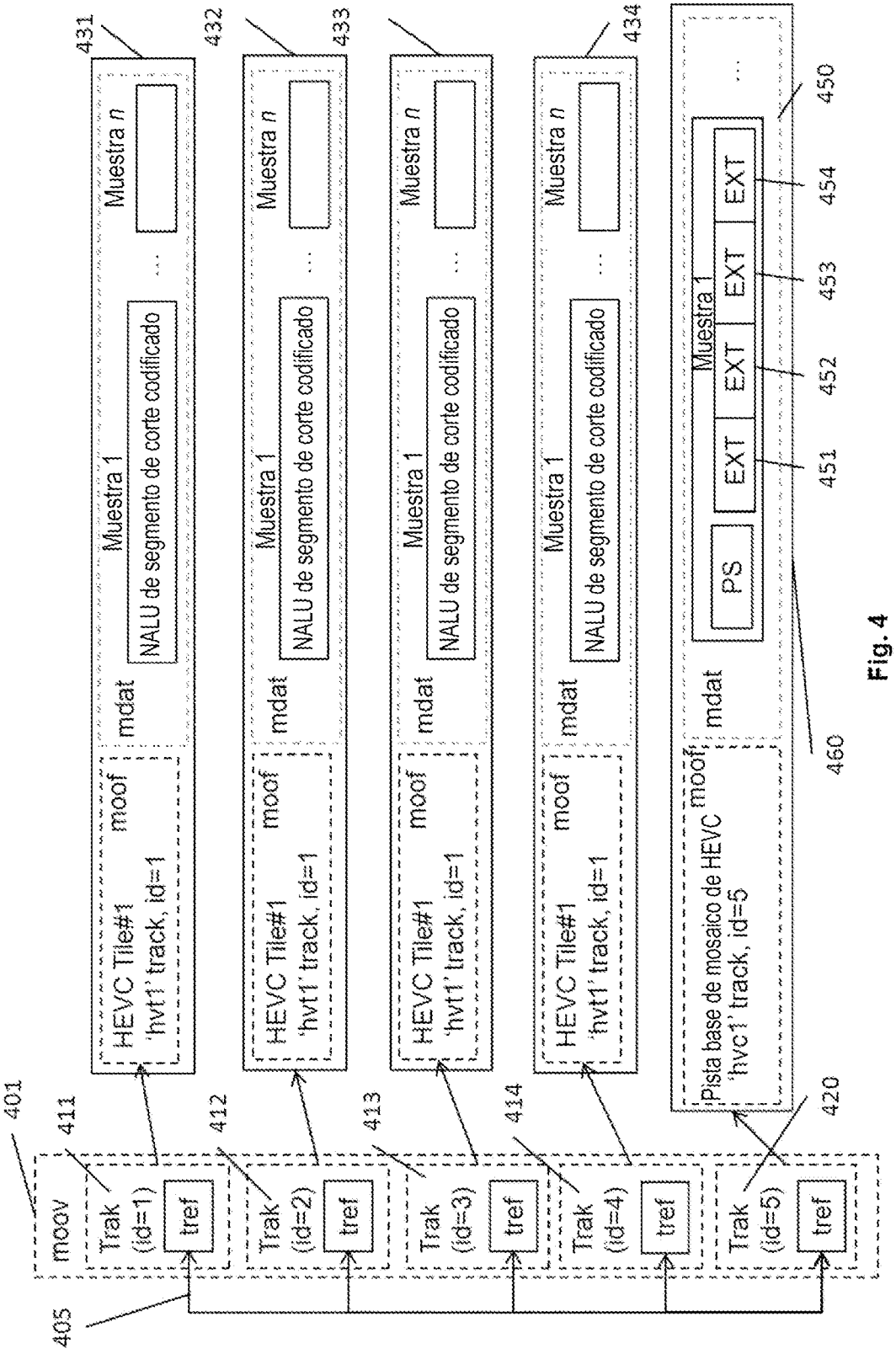
401
405
moov
411
Trak
(id=1)
tref
412
Trak
(id=2)
tref
413
Trak
(id=3)
tref
414
Trak
(id=4)
tref
Trak
(id=5)
tref
420
HEVC Tile#1
'hvt1' track, id=1
moof
mdat
Muestra 1
NALU de segmento de corte codificado
...
Muestra n
431
432
433
434
Pista base de mosaico de HEVC
'hvc1' track, id=5
PS
EXT
451
452
453
454
460
450

Fig. 4

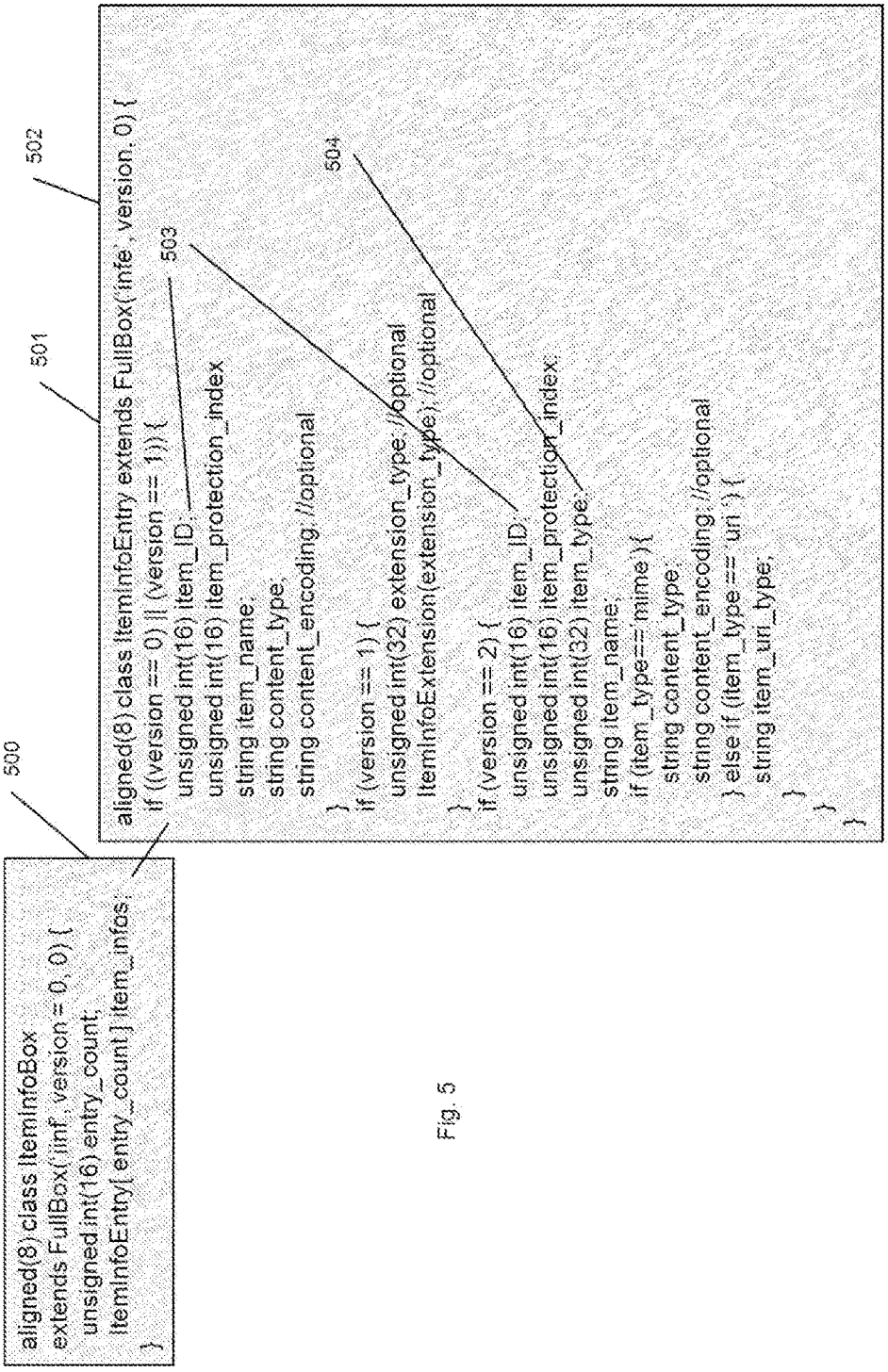
500
501
502
503
504
aligned(8) class ItemInfoBox
extends FullBox('iinf', version = 0, 0) {
unsigned int(16) entry_count;
ItemInfoEntry[entry_count] item_infos;
}
aligned(8) class ItemInfoEntry extends FullBox('infe', version, 0) {
if ((version == 0) || (version == 1)) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index
string item_name;
string content_type;
string content_encoding; //optional
}
if (version == 1) {
unsigned int(32) extension_type; //optional
ItemInfoExtension(extension_type); //optional
}
if (version == 2) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index;
unsigned int(32) item_type;
string item_name;
if (item_type=='mime'){
string content_type;
string content_encoding; //optional
} else if (item_type == 'uri ') {
string item_uri_type;
}
}
}

Fig. 5

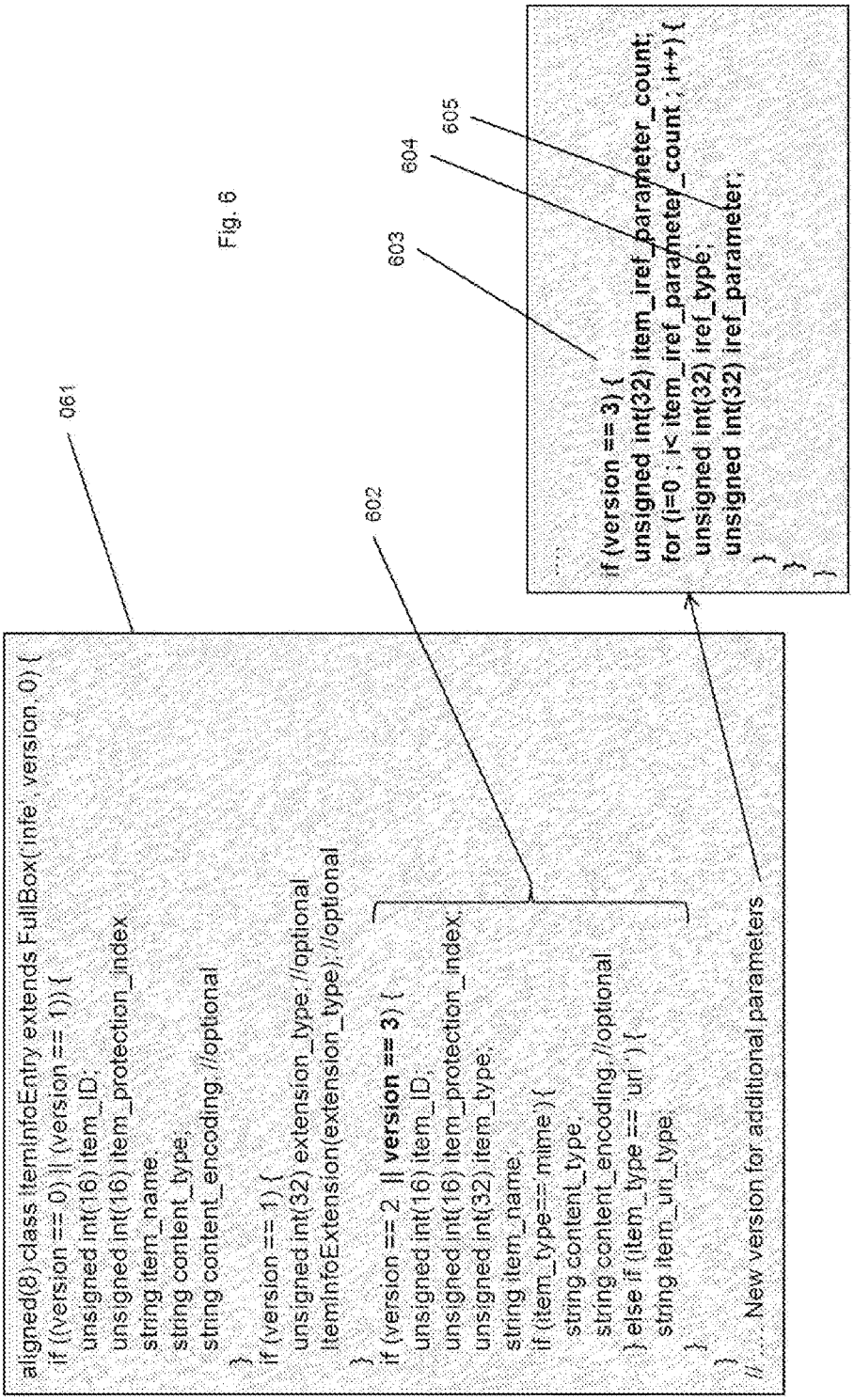
aligned(8) class iteminfoEntry extends FullBox('infe', version, 0) {
if ((version == 0) || (version == 1)) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index
string item_name;
string content_type;
string content_encoding; //optional
}
if (version == 1) {
unsigned int(32) extension_type; //optional
ItemInfoExtension(extension_type); //optional
}
if (version == 2 || version == 3) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index;
unsigned int(32) item_type;
string item_name;
if (item_type== 'mime'){
string content_type;
string content_encoding; //optional
} else if (item_type == 'uri ') {
string item_uri_type;
}
}
// ... New version for additional parameters
....
if (version == 3) {
unsigned int(32) item_iref_parameter_count;
for (i=0 ; i< item_iref_parameter_count ; i++) {
unsigned int(32) iref_type;
unsigned int(32) iref_parameter;
}
}
}
061
602
603
604
605

Fig. 6

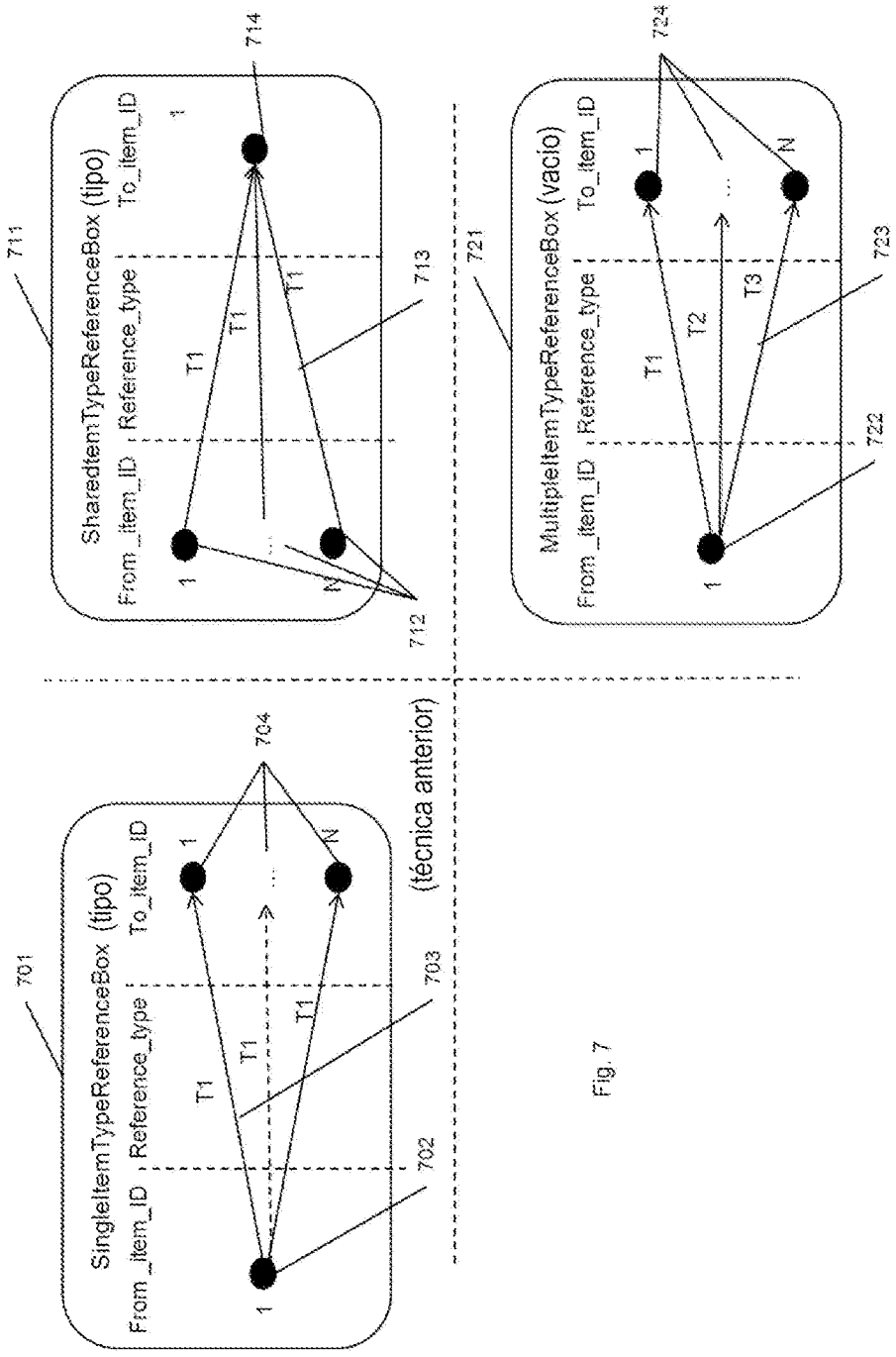
SingleItemTypeReferenceBox (tipo)
From_item_ID
Reference_type
To_item_ID
701
702
703
704
T1
1
N
(técnica anterior)
SharedItemTypeReferenceBox (tipo)
711
712
713
714
MultipleItemTypeReferenceBox (vacío)
721
722
723
724
T2
T3

Fig. 7

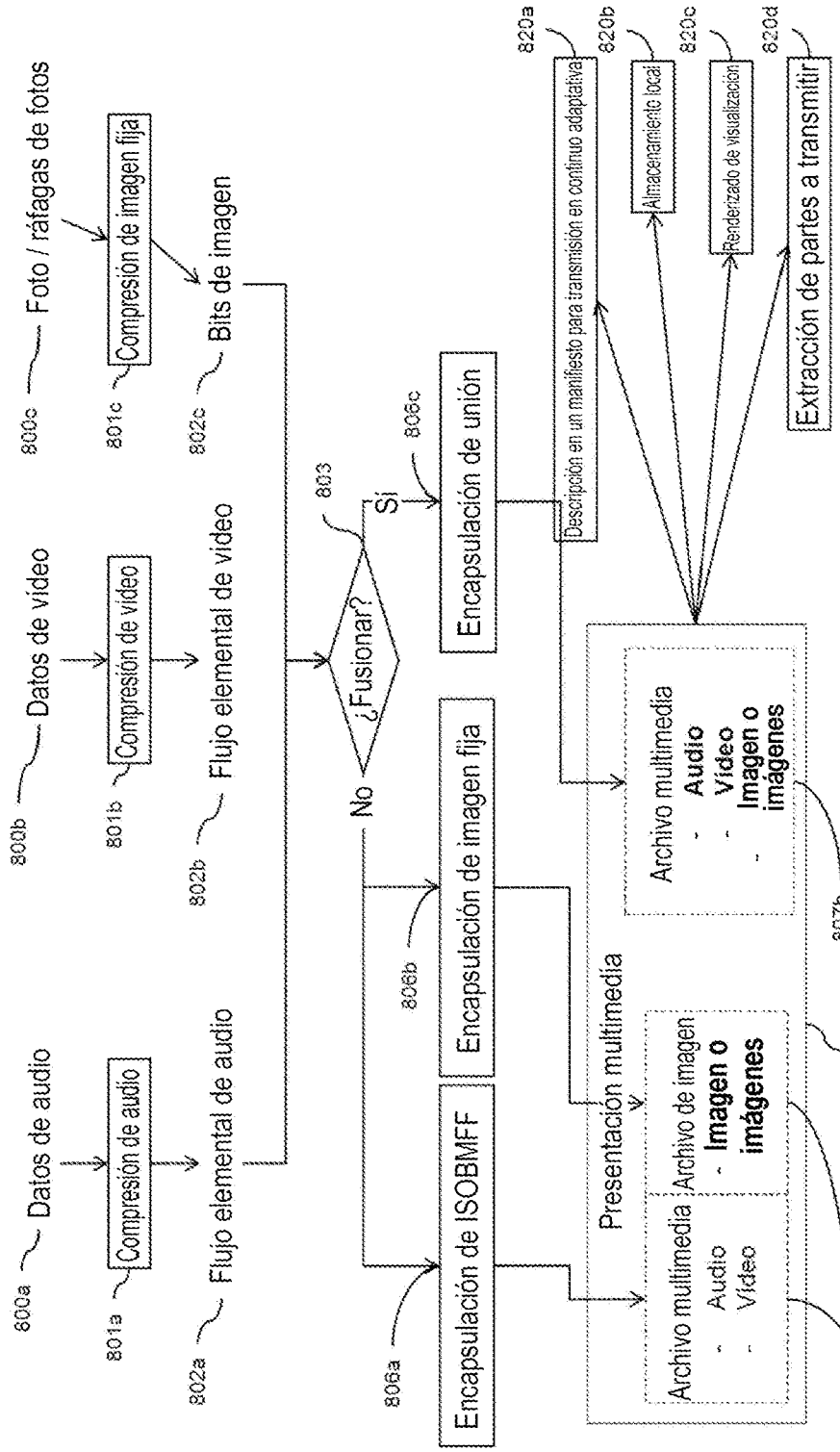
800a
Datos de audio
801a
Compresión de audio
802a
Flujo elemental de audio
800b
Datos de vídeo
801b
Compresión de vídeo
802b
Flujo elemental de vídeo
800c
Foto / ráfagas de fotos
801c
Compresión de imagen fija
802c
Bits de imagen
803
¿Fusionar?
No
Sí
806a
Encapsulación de ISOBMFF
806b
Encapsulación de imagen fija
806c
Encapsulación de unión
Presentación multimedia
Archivo multimedia
- Audio
- Vídeo
Archivo de imagen
- Imagen o imágenes
807
807a
Archivo multimedia
- Audio
- Vídeo
- Imagen o imágenes
807b
820a
Descripción en un manifiesto para transmisión en continuo adaptativa
820b
Almacenamiento local
820c
Renderizado de visualización
820d
Extracción de partes a transmitir

Fig. 8

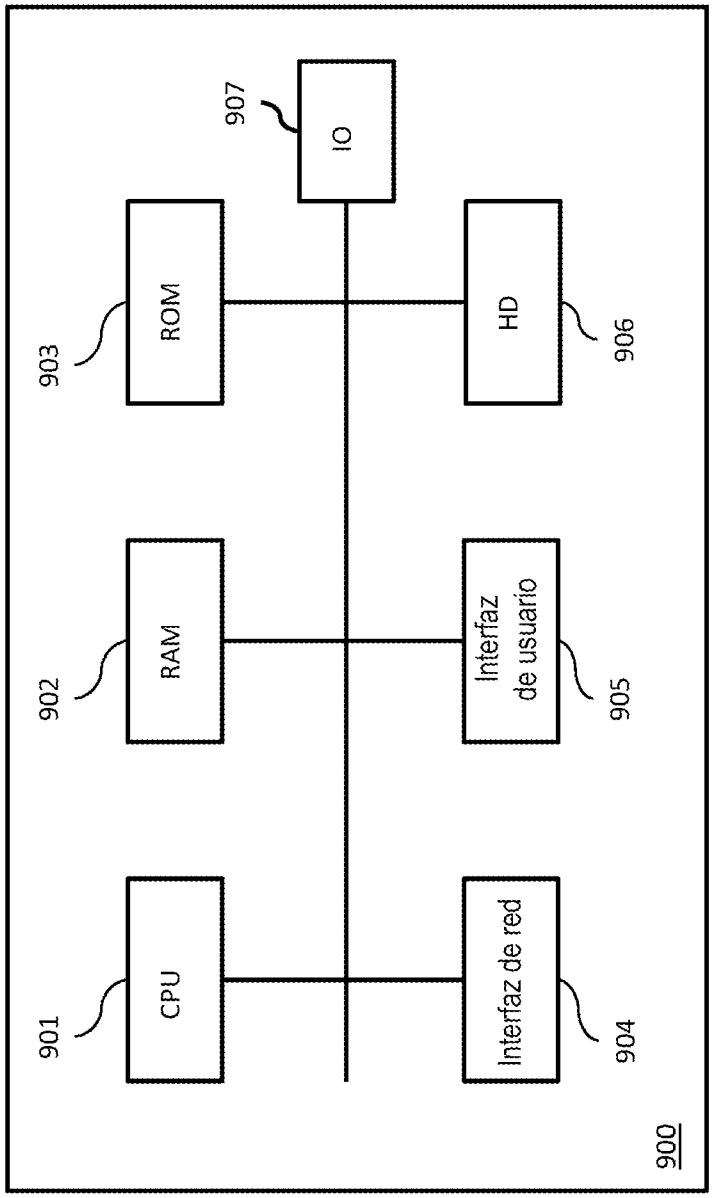
907
IO
903
ROM
HD
906
902
RAM
Interfaz
de usuario
905
901
CPU
Interfaz de red
904
900

Figura 9