CAPÍTULO 2: VÍDEO COMPRIMIDO

2. VÍDEO COMPRIMIDO

2.1. Antecedentes

La cantidad de información que diariamente manejamos va en aumento. El uso de conexiones de Internet de gran ancho de banda, la capacidad de almacenamiento de los discos duros, memorias flash y soportes ópticos es cada vez mayor. El precio por bit transmitido o almacenado es cada vez menor y, aún así, la compresión se hace necesaria para facilitar el almacenamiento y transmisión de la información.

La compresión de vídeo tiene dos importantes ventajas: hace posible utilizar vídeo digital en medios que no soportarían información sin comprimir (los anchos de banda de Internet son insuficientes para manejar vídeo sin comprimir en tiempo real y un DVD sólo puede almacenar unos segundos de éste) y hace un uso más eficiente de los recursos de transmisión y almacenamiento de la información. En resumen, la compresión de vídeo hace que la difusión de éste sea más sencilla y rápida, sin una pérdida apreciable de calidad.

La mayoría de las técnicas de compresión se basan en la compresión con pérdidas, donde se consigue una mayor compresión a costa de la calidad de la señal reconstruida. El objetivo de un algoritmo para la compresión de vídeo es maximizar la eficiencia de la compresión a la vez que minimizar la distorsión introducida durante ésta.

El grupo MPEG (Moving Picture Expert Group) es un grupo de trabajo de la ISO (International Standardization Organization) y del IEC (International Electrotechnical Commision). El cometido de este grupo es el de desarrollar estándares para la compresión, procesamiento y representación de señales de audio y video, actividad que ha desempeñado desde 1991.

El primero de la serie de estándares fue MPEG-1, creado con la intención de hacer posible el almacenamiento de vídeo en discos ópticos. El siguiente fue MPEG-2, el estándar audiovisual más extendido en aplicaciones multimedia, usado en la televisión digital, DVDs y ordenadores personales. Con un formato más potente que MPEG-1, es capaz de alcanzar niveles de compresión muy elevados, manteniendo, no obstante, una buena calidad de vídeo.

MPEG-4 Visual (ISO 14496-2 Part 2: ”Coding of Audio-Visual Object”) es el sucesor de MPEG-2. Este estándar supera al anterior en eficiencia de compresión, consiguiendo reducir la información a transmitir manteniendo la misma calidad de imagen, y en flexibilidad, con un amplio abanico de aplicaciones. Esto lo ha conseguido de dos maneras: usando algoritmos de compresión más avanzados y proporcionando un conjunto de herramientas para manipular y codificar la información multimedia.

MPEG-4 Visual consiste en un modelo básico de codificador-descodificador junto con un número herramientas de codificación. El modelo básico es el de codificación híbrido DPCM/DCT (un modelo de códec que usa compensación de movimiento basado en bloques, transformada, cuantización y codificación de entropía) y las funciones adicionales posibilitan, entre otras cosas, un aumento de la eficiencia de la compresión, transmisión fiable, y codificación de una escena por objetos o modelos de animación de cara o cuerpo [14, 18].

2.2. Captura de vídeo

La codificación de vídeo es el proceso de comprimir y descomprimir la señal de vídeo digital. Para comprender cómo funciona la codificación de vídeo es mejor examinar primero la estructura y características de las imágenes digitales y algunos conceptos de muestreo y calidad, para después pasar a describir las partes fundamentales de la compresión.

2.2.1. Muestreo de imágenes

Una imagen de vídeo típica está compuesta de una serie de objetos cada uno con su forma, textura, profundidad e iluminación. El color y el brillo de una escena de vídeo cambian a lo largo de ella, variando el nivel de intensidad. Las características de la imagen relevantes en la codificación de vídeo son espaciales (variación de la textura en la escena, número y forma de los objetos, etc.) y temporales (movimiento de los objetos, cambios en la iluminación, movimiento de la cámara, etc.).

Para representar una imagen en forma digital se necesita muestrearla de forma espacial y de forma temporal. El muestreo espacial se suele hacer superponiendo un rectángulo con los puntos de muestreo, como puede verse en la ilustración 1. La imagen muestreada se reconstruirá representado cada muestra como un elemento de imagen o píxel. La calidad visual de la imagen reconstruida dependerá de la cantidad de muestras tomadas.

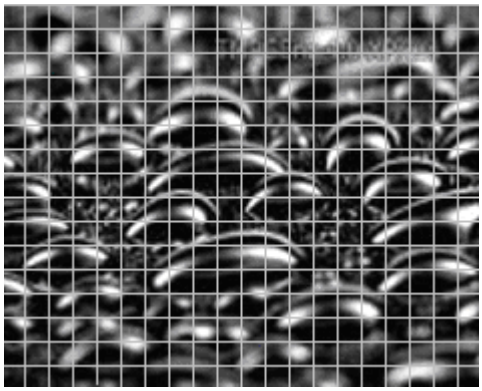

Ilustración 1: Muestreo espacial

Las muestras temporales son imágenes tomadas cada cierto tiempo, como se puede ver en la ilustración 2. Al reproducir la serie de imágenes o cuadros se produce la sensación de movimiento. A mayor frecuencia de muestreo -más cuadros por segundo- se apreciará un movimiento más suave, pero requerirá de más muestras y por tanto de más información a transmitir. La frecuencia estándar es para televisión de 25 ó 30 cuadros por segundo.

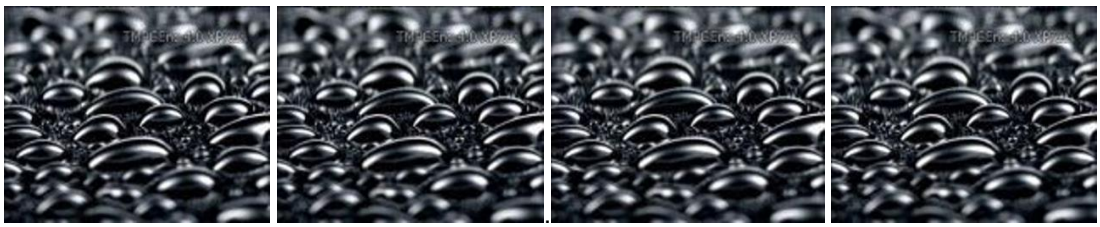

Ilustración 2: Muestreo temporal

Una señal de vídeo se puede muestrear como una serie completa de cuadros o imágenes (muestreo progresivo) o como una secuencia de campos entrelazados (muestreo entrelazado). En una secuencia de vídeo entrelazada la mitad de los datos del cuadro completo, un campo, se muestrea cada cierto tiempo. Un campo consiste en las líneas pares o impares de un cuadro completo, y una secuencia de vídeo entrelazada contiene una serie de campos, cada uno representando la mitad de la información del cuadro completo. La principal ventaja de este método de muestreo es que permite mandar el doble de campos por segundo (en televisión 50 ó 60 campos por segundo), dando una mejor sensación de movimiento. En la ilustración 3 se puede ver cómo es el muestreo de una secuencia de vídeo entrelazada:

Ilustración 3: Secuencia de vídeo entrelazada.

Tras el proceso de muestreo del vídeo se tienen una serie de muestras espacio-temporales (correspondientes a un píxel o elemento de la imagen), representándose cada una por una serie de números que describen el brillo o luminancia y el color de la muestra.

2.2.2. Espacio de color

La mayoría de las aplicaciones de vídeo utilizan pantallas de color, por lo que se necesita un mecanismo para capturar y representar la información de color. Una imagen monocroma necesita sólo un número para indicar el brillo o luminancia de cada muestra espacial. Las imágenes de color, por otro lado, requieren al menos tres números para poder reproducir el color de forma precisa. El método elegido para representar tanto la luminancia como el color se conoce como espacio de color.

En el espacio de color RGB, una muestra de una imagen en color se representa con tres números que indican la proporción relativa de rojo, verde y azul, los tres colores primarios aditivos de la luz. Cualquier color puede representarse combinando éstos en distintas proporciones. Este espacio de color encaja muy bien en la captura y representación de imágenes, ya que las pantallas contienen elementos de imagen para cada uno de los tres colores. Sin embargo, hay que tener en cuenta otros factores al elegir el espacio de color.

En el espacio de color RGB los tres colores tienen igual importancia y se almacenan con la misma resolución. Sin embargo, teniendo en cuenta las deficiencias del sistema visual humano, que es menos sensible al color que a la luminancia, es posible representar una imagen de color de forma más eficiente. Esto lo hace el espacio de color YC_rC_b, que reduce el ancho de banda de la croma respecto a la luminancia, sin que la pérdida de la calidad sea apreciable.

Esta forma de representación del color y sus variantes (como YUV) es una manera muy aceptada de representar las imágenes de color. *Y* es la componente de luminancia y se puede calcular como una media ponderada de las componentes *R*, *G* y *B*:

$$Y = kr \cdot R + kg \cdot G + kb \cdot B$$

La información de color puede ser representada mediante diferencias de color, croma o crominancia, donde cada componente de croma es la diferencia entre *R*, *G* o *B* y la luminancia *Y*:

$$Cb = B - Y$$

$$Cr = R - Y$$

$$Cg = G - Y$$

La descripción completa de una imagen de color se da con la luminancia y con dos de las diferencias de color, ya que la tercera componente se puede obtener a partir de las otras

dos. En el espacio de color YC_rC_b sólo se transmiten las componentes de color azul y roja, además de la luminancia. Este espacio utiliza una menor resolución para la croma que para la luminancia, lo que reduce la cantidad de información que se necesita para representar las componentes de crominancia sin disminuir la calidad, de forma que un observador no aprecia una diferencia entre una imagen RGB y una YCbCr. Este espacio de color se convierte, por tanto, en una forma efectiva y sencilla de comprimir la información [5, 18].

2.2.3. Formatos de muestreo

Los sistemas de vídeo transmiten los datos de la imagen con tres componentes: una que representa el brillo, la luminancia, y dos componentes que representan el color. Este esquema explota las deficiencias del sistema visual humano, ya que la sensibilidad de éste al color es mucho menor que a la luminancia. Mientras la luminancia se mande con resolución máxima, el color puede submuestrearse manteniendo una buena calidad en la imagen.

El muestreo se expresa con tres números enteros separados por dos puntos. La relación entre los dígitos denota el grado de submuestreo vertical y horizontal. El primer dígito indica el número relativo de muestras luminancia; el segundo indica el nivel de submuestreo de las componentes de croma y el tercero, si es cero, indica un submuestreo de dos en las componentes de croma. MPEG-4 Visual soporta tres tipos de formatos de muestreo: 4:4:4, 4:2:2 y 4:2:0. En el primero, las tres componentes (Y, C_b y C_r) se envían con la misma resolución, no hay submuestreo de croma y, por tanto, se utiliza en aplicaciones de vídeo de muy alta calidad de color. En el muestreo 4:2:2, las componentes de croma tienen la misma resolución vertical que la luminancia pero la mitad de la resolución horizontal. Este tipo de muestreo se utiliza en imágenes con reproducción de color de calidad.

En el formato de muestreo más usado, 4:2:0, las componentes de croma tienen la mitad de resolución horizontal y la mitad de resolución vertical que la luminancia. Este tipo se utiliza en aplicaciones como en la vídeoconferencia, en la televisión digital y el en almacenamiento en DVD. Al tener cada componente de croma la cuarta parte de muestras que la luminancia, el vídeo en formato 4:2:0 necesita exactamente la mitad de muestras que el video RGB [8, 17,18]. Las ilustraciones 4,5 y 6 muestran los tres formatos de muestreo:

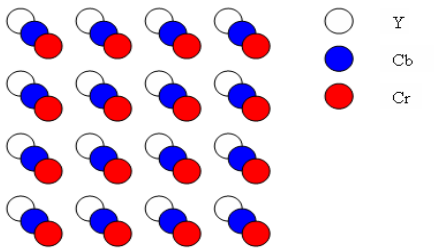

Ilustración 4: Muestreo 4:4:4

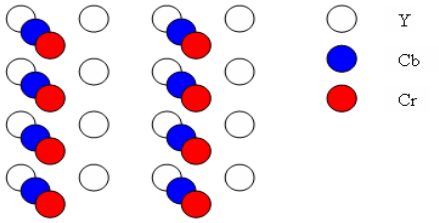

Ilustración 5: Muestreo 4:2:2.

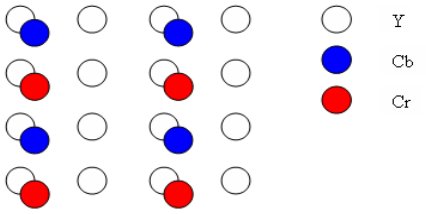

Ilustración 6: Muestreo 4:2:0.

2.2.4. Formato de vídeo

El estándar de compresión de video puede comprimir una amplia variedad de formatos de video. En la práctica se suele capturar o convertir el vídeo en varios formatos intermedios antes de la compresión y la transmisión. La elección de la resolución depende de la aplicación y la capacidad de almacenamiento o transmisión. El formato CIF (Common Intermediate Format) es la base de los formatos que más se utilizan y tiene 352 píxeles por línea y 288 líneas. También se utilizan el formato 4CIF (704x576) que es el más apropiado para la televisión estándar y el vídeo DVD; CIF y QCIF (Quarter CIF, 176x144) se usan en aplicaciones de vídeo conferencia; QCIF o SQCIF (Sub-CIF, 128x96) se usan en aplicaciones multimedia de móviles, donde la resolución de la pantalla y la tasa de transmisión son pequeñas. Estos son los 5 formatos de vídeo estandarizados, pero es posible utilizar otro formato definido por el consumidor [22].

Se ha visto pues que con el muestreo de una señal analógica se obtiene una señal digital que proporciona más calidad de imagen y precisión, y que es compatible con otros medios digitales y sistemas de transmisión, pero que, normalmente, ocupa un gran ancho de banda. El siguiente punto introduce los conceptos básicos de la compresión de video, necesaria para optimizar las capacidades de almacenamiento y transmisión.

2.3. Codificación de vídeo

La compresión de video es el proceso de compactar o condensar una secuencia digital de video en un número pequeño de bits. La compresión se consigue eliminando la redundancia en los datos, es decir, eliminando los datos que no son necesarios para una reproducción fiel del vídeo original. Muchos tipos de datos contienen redundancia estadística,

y se pueden comprimir de forma efectiva y sin pérdidas, de forma que la señal reconstruida sea una copia perfecta de la original. Con esta compresión sin pérdidas se consigue reducir la cantidad de información tres o cuatro veces, por lo que se hace necesario el empleo de una compresión con pérdidas a fin de obtener unos niveles de compresión mayores.

En un sistema de compresión con pérdidas, la información descomprimida no es idéntica a la original, y se pueden conseguir niveles de compresión mayores a costa de una pérdida de calidad visual. Los sistemas de compresión con pérdidas se basan en eliminar la redundancia, es decir, los elementos de la imagen o de la secuencia de vídeo que se pueden eliminar sin afectar a la calidad visual que percibe el espectador.

Existen dos tipos de redundancia que explotan la mayoría de los métodos de compresión para conseguir que ésta sea eficiente: la redundancia temporal y la espacial. En el dominio temporal, hay normalmente una alta correlación entre cuadros capturados en instantes de tiempo consecutivos, especialmente si la frecuencia de muestreo temporal es alta. En el dominio espacial, hay normalmente un gran parecido entre píxeles que se encuentran en posiciones próximas de forma que las muestras adyacentes son muy similares.

El estándar MPEG-4 Visual define un modelo de códec que usa compensación de movimiento basada en bloques, transformada, cuantización y codificación de entropía. En la ilustración 7 se puede ver el modelo básico que sigue este estándar:

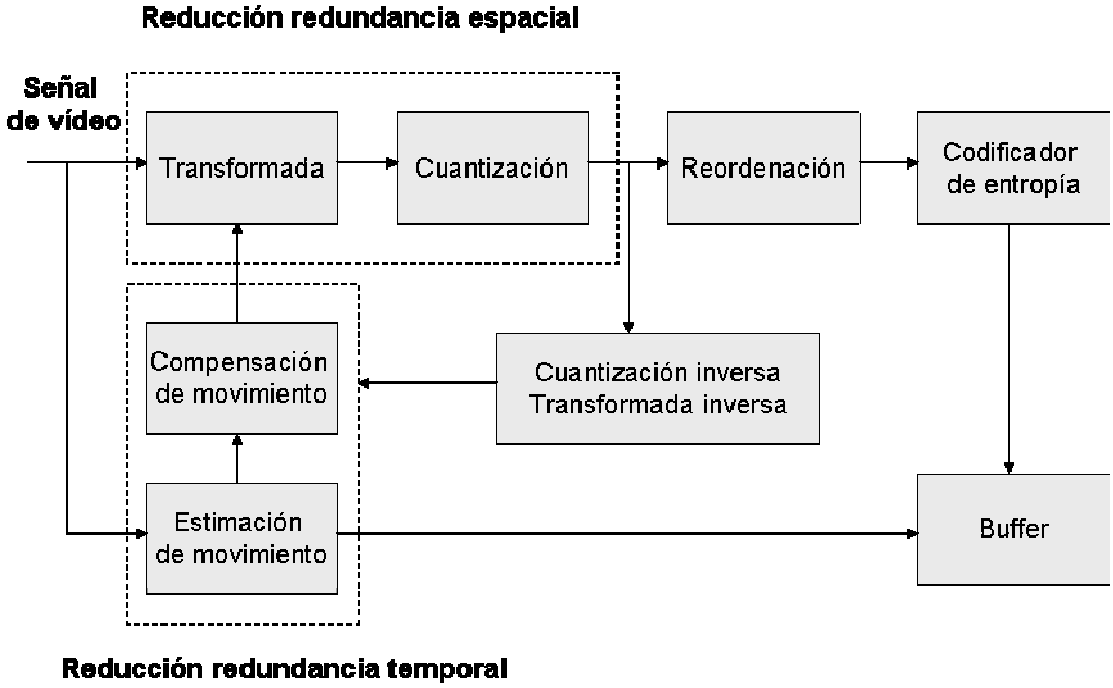

Ilustración 7: Modelo básico de un codificador

El modelo de códec básico transforma la secuencia de video original en otra que tiene el menor número de bits posible, pero que al decodificarla debe ser lo más fiel posible a la original. Estos dos objetivos, conseguir una alta eficiencia en la compresión y una buena calidad, son incompatibles, ya que niveles de compresión mayores suponen una menor calidad de la imagen reconstruida.

Un codificador de video consiste en tres unidades funcionales básicas: el modelo temporal, el modelo espacial y el codificador de entropía. La entrada al modelo temporal es la secuencia de video original. Este modelo intenta reducir la redundancia temporal explotando el parecido entre cuadros consecutivos, normalmente usando una predicción que se forma del cuadro actual y compensando las diferencias entre la predicción y el original (predicción de compensación de movimiento). La salida del modelo temporal es un cuadro residual (obtenido restando la predicción del cuadro actual) y un conjunto de los parámetros del modelo, normalmente vectores de movimiento describiendo cómo se compensó el movimiento.

El cuadro residual es la entrada al modelo espacial que utiliza el parecido entre muestras adyacentes para reducir la redundancia espacial, aplicando una transformada a las muestras residuales y cuantificando los resultados. La salida del modelo espacial es un conjunto reducido de coeficientes cuantificados de la transformada.

Los parámetros del modelo temporal, normalmente vectores de movimiento, y del modelo espacial, los coeficientes, se comprimen en el codificador de entropía. Éste elimina la redundancia estadística de los datos, (por ejemplo, asignando un código corto a los coeficientes o vectores de movimiento que se repiten mucho) y produce una secuencia de bits comprimida para transmitir o almacenar. La secuencia de video comprimida consiste, por tanto, en los parámetros de los vectores de movimiento codificados, los coeficientes residuales codificados e información de cabecera.

El decodificador reconstruye cada cuadro a partir de la secuencia de video comprimida. Los coeficientes y los vectores de movimiento se decodifican en el decodificador de entropía para después en el modelo temporal reconstruir una versión del cuadro residual. Éste se utiliza junto con los vectores de movimiento y con uno o varios cuadros anteriores o posteriores para crear la predicción del cuadro actual y sumársela al residual para reconstruir el original [3, 18].

2.3.1. Redundancia temporal

El objetivo del modelo temporal es reducir la redundancia entre cuadros consecutivos, formando una predicción y restando ésta al cuadro actual. La salida de este proceso es un

cuadro residual, y cuanto más exacta sea la predicción, menos energía contendrá el residual. El cuadro residual se codifica y se envía al decodificador que recrea la predicción, añade el residual decodificado y reconstruye el cuadro original. La predicción se forma a partir de uno o más cuadros anteriores o posteriores (cuadros de referencia). La exactitud de la predicción puede mejorarse compensando el movimiento entre el cuadro de referencia y el cuadro actual.

En la mayoría de los estándares de codificación de vídeo, incluyendo MPEG-1, MPEG-2 y MPEG-4 Visual, el macrobloque, una porción del cuadro de tamaño 16x16 píxeles, es la unidad básica de compensación de movimiento. Para un formato de muestreo de 4:2:0, el macrobloque se puede representar de la siguiente forma: una región de 16x16 píxeles corresponde a 256 muestras de luminancia, divididas en 4 bloques de 8x8; 64 muestras de croma azul, un bloque de 8x8 y 64 de croma roja en otro bloque de 8x8. MPEG-4 Visual procesa cada cuadro en unidades de macrobloques. La ilustración 8 lo muestra:

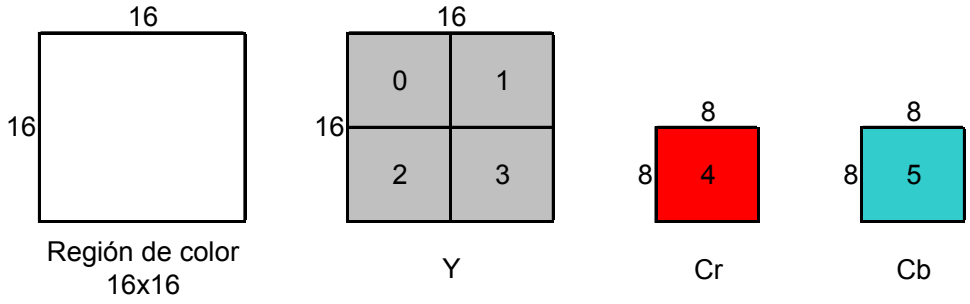

Ilustración 8: Macrobloque 4:2:0

Un método muy práctico y utilizado para la compensación de movimiento es compensar éste en regiones rectangulares o bloques del cuadro actual. En MPEG-4 Visual, el proceso se hace para cada macrobloque. En primer lugar, se procede a la estimación del movimiento, buscando en un área de búsqueda del cuadro de referencia (pasado o futuro, previamente codificado y transmitido), un bloque que se parezca más al macrobloque actual. Esto se hace comparando el macrobloque del cuadro actual con cada uno de los bloques del área de búsqueda, centrada normalmente en la posición actual del macrobloque. Un criterio común de elección de la región es la energía del residual producido al restar la región candidata a la original. El bloque elegido se etiqueta como “best-match”. La ilustración 10 muestra este proceso:

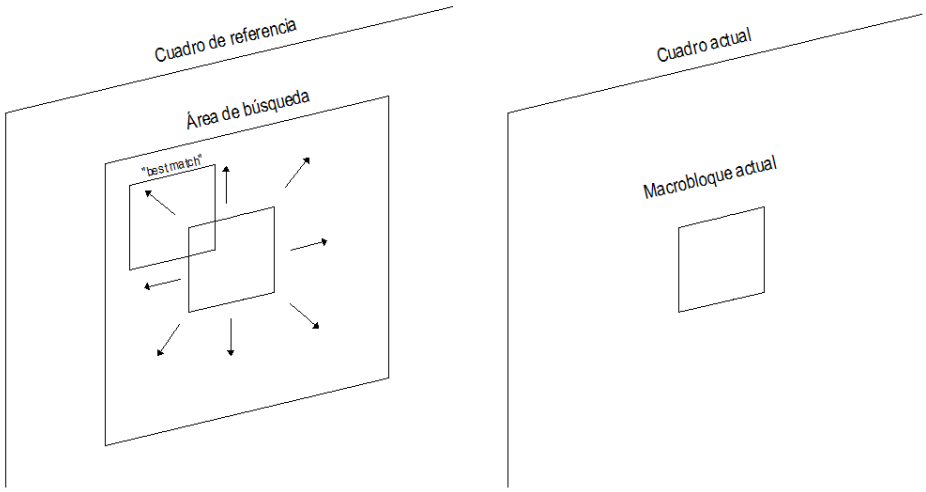

Ilustración 9: Estimación de movimiento

La región elegida se convierte en el "predictor" para el bloque actual y se le resta para formar un macrobloque residual, de luminancia y croma, que se codifica y transmite junto con el vector de movimiento, que describe la posición relativa del bloque elegido como "best match". Dentro del codificador, el residual se codifica, decodifica y se añade a la región elegida para formar un macrobloque reconstruido que se guarda como referencia para una posterior predicción de compensación de movimiento. Es necesario utilizar el residual decodificado para asegurar que el codificador y el decodificador utilizan la misma referencia para la compensación de movimiento.

En el decodificador se utiliza el vector de movimiento recibido para recrear la región "predictor" y decodifica el bloque residual, lo añade al "predictor" y construye una versión del bloque original.

La compensación de movimiento basada en bloques es muy utilizada por varias razones: es sencilla y tratable de forma computacional, encaja bien en la codificación de cuadros rectangulares y con transformadas de imagen basadas en bloques (como la DCT, que se describirá más tarde). En general, proporciona un modelo temporal razonablemente efectivo para muchas secuencias de vídeo. Sin embargo, también tiene desventajas. Por ejemplo, los objetos se moverán un número fraccional de posiciones de píxel y muchos modos de movimiento de objetos son difíciles de compensar como es el caso de rotaciones u objetos deformables. A pesar de estas desventajas, la compensación de movimiento basada en bloques es la base del modelo temporal utilizado en todos los estándares de codificación de video actuales.

Existen muchas variantes al modelo básico de estimación y compensación de movimiento. El cuadro de referencia puede ser anterior en la secuencia de imágenes,

posterior o una combinación de ambos. Si se elige un cuadro de referencia futuro, se necesita codificar antes la referencia que el cuadro actual. Si hay una gran diferencia entre el cuadro actual y el de referencia como, por ejemplo, un cambio de plano, puede resultar más eficiente codificar el macrobloque sin compensación de movimiento, de forma que el codificador pueda elegir entre codificar cada macrobloque en modo *intra* (codificar sin compensación de movimiento) o *inter* (codificar con compensación de movimiento). Se puede utilizar un tamaño de bloque variable para la estimación y compensación de movimiento y usar interpolación para crear muestras en posiciones subpixel, de forma que aumente la precisión de la estimación [3, 11, 18].

2.3.2. Redundancia espacial

Una imagen de vídeo real consiste en una matriz de muestras. Las imágenes reales son difíciles de comprimir por la alta correlación entre muestras adyacentes. Sin embargo, la imagen residual que queda tras la compensación de movimiento tiene una función de correlación que decae rápido a medida que crece la distancia, indicando que las muestras adyacentes están poco correlacionadas. Una compensación de movimiento eficiente reduce la correlación local en el residual, lo que lo hace más fácil comprimir que la imagen original. Esto se puede apreciar en la ilustración 10:

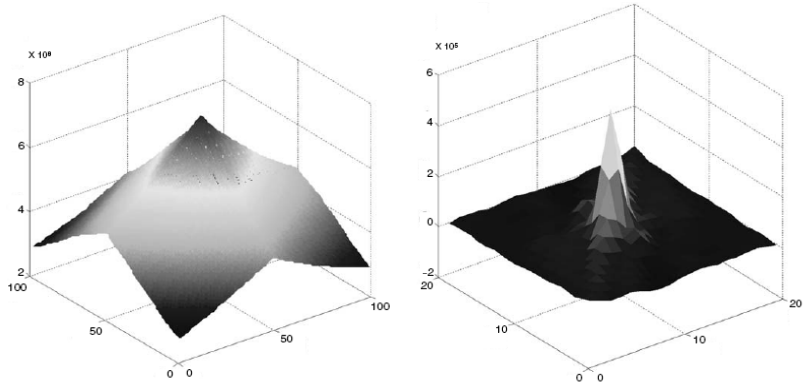

Ilustración 10: Correlación en la imagen original y en el residual.

La función del modelo espacial es reducir los datos de la imagen todo lo posible para que esté en una forma que pueda ser eficientemente comprimida en un codificador de entropía. Los modelos de imagen más comunes tienen tres componentes principales: transformación, que compacta la información; cuantización, que reduce la precisión de los datos transformados, y reordenación, que agrupa los valores más significativos.

De una forma similar a la predicción temporal para reducir la cantidad de información, se puede hacer una predicción de la imagen de forma a partir de otras muestras de la misma imagen. La codificación predictiva se ha usado desde los primeros algoritmos de codificación y se suele llamar DPCM (Diferencial Pulse Code Modulation).

En la ilustración 10 se puede ver un píxel X que ha de codificarse. Si el cuadro se procesa en orden matricial (de izquierda a derecha y de arriba a abajo), los píxeles A, B y C (píxeles vecinos en la fila anterior y en la actual) están disponibles tanto en el codificador como en el decodificador. El codificador forma una predicción de X basada en alguna combinación de píxeles previamente codificados, la resta de X y codifica el residual. El decodificador forma la misma predicción y añade el residual a ésta para reconstruir el píxel.

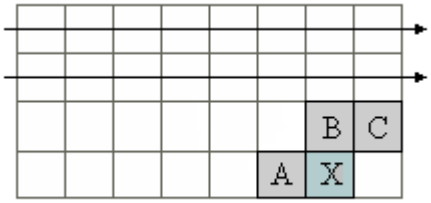

Ilustración 11: Codificación predictiva

Si el proceso de decodificación es con pérdidas, puede que los píxeles A, B y C reconstruidos no sean idénticos a los originales y se vaya produciendo un error acumulativo. En ese caso, el codificador debería decodificar los residuales, reconstruir los píxeles y usar éstos para formar la predicción minimizando así los errores.

La eficiencia de la compresión conseguida de esta manera dependerá de la exactitud de la predicción. Si la predicción es buena, entonces el residual será pequeño. No obstante, es complicado elegir un "predictor" que funcione bien para todas las áreas de una imagen. Un mejor resultado se puede obtener adaptando el "predictor" dependiendo de las estadísticas de la imagen, siendo necesario que sea especificado por el codificador, de forma que la eficiencia conseguida con este método compense la información extra que habrá que enviar.

2.3.2.1. Transformada

El objetivo de la capa de transformada en un codificador de vídeo es convertir el residual que queda tras la compensación de movimiento en otro dominio: el de la transformada. La transformada elegida deberá compactar los datos en un número pequeño de coeficientes, ser reversible y tratable de forma computacional. La mayoría de las transformadas propuestas para codificación de vídeo se pueden clasificar en dos categorías: basadas en bloques y basadas en imagen.

Las transformadas basadas en bloques operan sobre bloques NxN de la imagen o sobre muestras residuales y por tanto la imagen se procesa en unidades de bloque. Estas transformadas requieren poca memoria y encajan bien con los algoritmos de compresión con compensación de movimiento por bloques, pero la correlación entre los bordes de los bloques no se elimina y esto resulta apreciable.

Por el contrario, las transformadas basadas en imágenes transforman el cuadro entero o una sección grande de éste. Estas transformadas funcionan mejor que las primeras pero necesitan de mayor capacidad de memoria por procesar la imagen completa como una unidad y no se complementa adecuadamente con la compensación de movimiento basada en bloques. MPEG-4 Visual soporta dos tipos de transformadas: la DCT (Discrete Cosine Transform), basada en bloques, y la DWT (Discrete Wavelet Transform), basada en imagen, que se describen a continuación.

Transformada DCT: La transformada DCT opera sobre un bloque de NxN muestras, para obtener un nuevo bloque Y de NxN coeficientes. La operación se puede describir mediante una matriz A, de forma que la transformación hacia delante sería:

$$Y = A \cdot X \cdot A^T$$

Y la transformada inversa:

$$X = A^T \cdot Y \cdot A$$

donde X es una matriz de muestras. Los elementos de A son:

$$A_{ij} = C_i \cdot \cos \frac{(2j+1)i\pi}{2N}$$

donde:

$$C_i = \sqrt{\frac{1}{N}} (i = 0) \quad C_i = \sqrt{\frac{2}{N}} (i > 0)$$

En general los coeficientes se pueden expresar de la siguiente manera:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N}$$

El resultado de aplicar la transformada es un conjunto de NxN coeficientes que representan los datos del bloque de la imagen en el dominio DCT. Estos coeficientes pueden considerarse como el peso de una serie de patrones básicos. Estos patrones

para bloques de 4x4 ó 8x8 se muestran en las ilustraciones 12 y 13, y se componen de combinaciones de funciones de coseno horizontal y vertical. Cualquier bloque de imagen puede reconstruirse combinando cada patrón NxN multiplicado por el peso que indica el coeficiente.

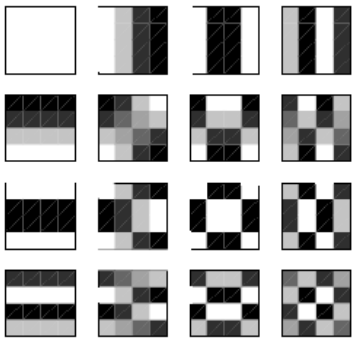

Ilustración 12: Patrón básico de una DCT 4x4

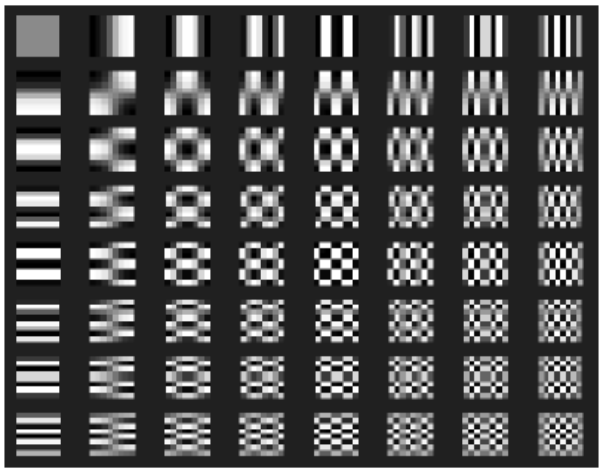

Ilustración 13: Patrón básico de una DCT 8x8

Hasta ahora no hemos conseguido una reducción de la información, en lugar de 16 valores de píxel tenemos 16 coeficientes. La utilidad de la transformada se aprecia cuando el bloque se reconstruye con sólo unos cuantos coeficientes. Al despreciar los coeficientes más insignificantes, normalmente mediante la cuantización, se puede construir la imagen con un número reducido de coeficientes a costa de una ligera pérdida de calidad [3, 10, 11,20].

Transformada DWT: el proceso básico de esta transformada consiste en filtrar una señal, que contiene N muestras, con un filtro paso de baja y con uno paso de alta y submuestrear la salida con un factor de dos. Ambas operaciones se hacen en dirección *x*. Con esto se ha dividido la señal en dos: una componente de baja frecuencia, L, y otra de alta frecuencia, H, cada una con N/2 muestras. Se filtran y

submuestrean de nuevo pero esta vez en dirección *y*. Se obtienen entonces 4 imágenes subbanda, una componente que representa el nivel medio (LL), y tres componentes que describen los detalles (LH, HL, HH). Todas estas componentes que pueden combinarse para recuperar la señal original. Se tiene la misma cantidad de información pero la nueva configuración hace más fácil una codificación eficiente.

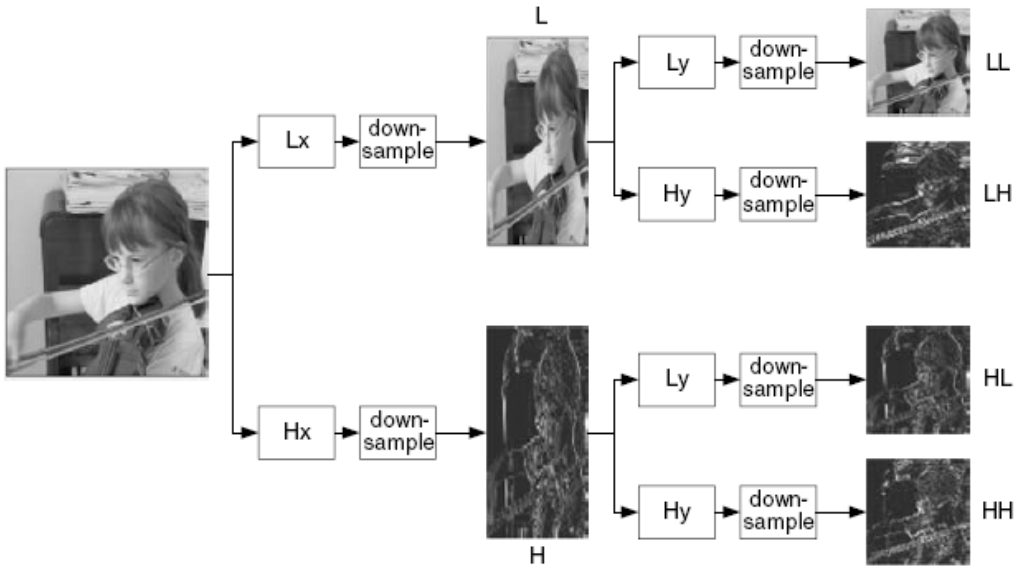

Ilustración 14: Transformada DWT

En aplicaciones de compresión de imágenes, se aplica de nuevo la descomposición wavelet a la subbanda LL, formando de nuevo otras cuatro subbandas. La imagen de la subbanda de menor frecuencia puede filtrarse de nuevo para crear un árbol de imágenes subbanda. En la ilustración 15 puede verse el resultado de aplicar dos procesos de descomposición:

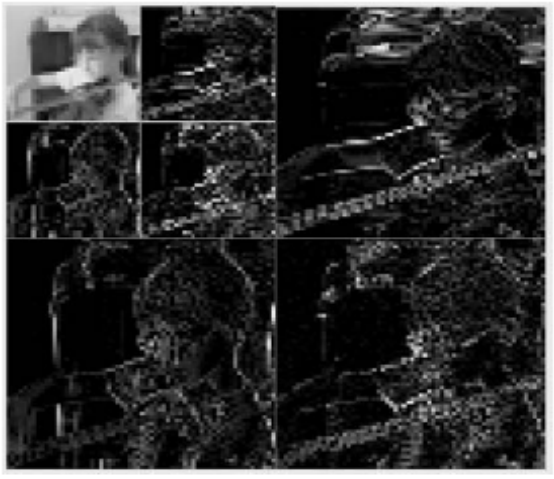

Ilustración 15: Dos niveles de descomposición de la DWT.

El objetivo de la DWT es transformar la imagen de una forma diferente de manera que el resultado obtenido carezca de los efectos de borde que aparecen en las transformadas basadas en bloques [3, 12].

2.3.2.2. Cuantización

Después de aplicar la transformada se tiene la misma cantidad de información, pero distribuida de manera que facilita la compresión. La cuantización mapea el rango de posibles valores de una señal *X* a un conjunto reducido de valores. Al reducir el número de posibles valores la señal se podrá representar con menos bits y, además, los valores insignificantes se eliminarán. Al cuantizar se divide cada valor por un número positivo no nulo (el paso de cuantización) y se redondea el resultado al entero más próximo. Los cuantizadores uniformes se definen de la siguiente forma:

$$FQ = round\left(\frac{X}{QP}\right)$$

$$Y = FQ \cdot QP$$

donde *QP* es el paso de cuantización. Los niveles cuantizados a la salida están separados *QP* unidades.

En la compresión de vídeo también puede utilizarse un cuantizador variable, que consiste en una matriz de NxN coeficientes que proporciona los valores de los pasos de cuantización, de forma que a coeficientes en distintas posiciones se les aplica un valor de *QP* diferente. El grado de cuantización aplicado a cada coeficiente depende de la visibilidad del ruido de cuantización que percibe un observador humano. En la práctica, la visibilidad se reduce para frecuencias más altas, para las que se pueden usar unos pasos de cuantización mayores que los utilizados para baja frecuencia.

La cuantización consigue reducir la precisión de los datos de la imagen después de aplicar la transformada, deshaciéndose de los valores menos significativos y cercanos a cero. El cuantizador se diseña para que todos los valores despreciables produzcan como salida valores nulos. Por tanto, a la salida del cuantizador tendremos un vector de coeficientes cuantizados con una gran mayoría de ceros [3, 22].

2.3.2.3. Reordenación

Los coeficientes de la transformada cuantizados deben codificarse de la forma más compacta posible, antes de su almacenamiento y transmisión. En un codificador de vídeo basado en la transformada, la salida del cuantizador es un vector que contiene algunos coeficientes distintos de cero junto con un gran número de valores nulos. Antes de la

codificación de entropía se reordenan los coeficientes distintos de cero y se representan de otra manera el resto de valores. La forma de reordenar estos coeficientes dependerá del tipo de transformada:

Transformada DCT. Los coeficientes significativos obtenidos al aplicar esta transformada sobre un bloque de una imagen son normalmente los de baja frecuencia, cercanos al coeficiente DC, de la posición (0,0). La ordenación óptima depende de la distribución de estos coeficientes, soliendo ser en zig-zag para un bloque de imagen típico. Comenzando por el coeficiente DC, el de la esquina superior izquierda, y seguido por los coeficientes AC, cada coeficiente cuantizado se copia en un vector unidimensional en orden. Los coeficientes nulos se agrupan al principio del vector reordenado, seguido de una larga secuencia de ceros.

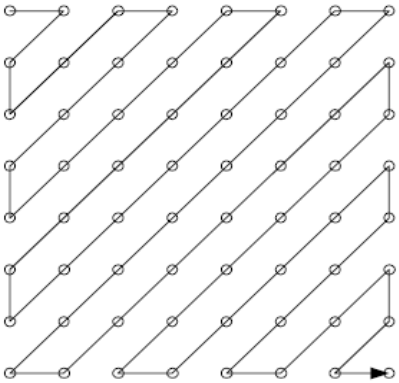

Ilustración 16: Reordenación en zig-zag.

Tras el proceso de reordenación de coeficientes normalmente se tienen uno o más grupos de coeficientes nulos. Éstos podrían ser representados de una forma más compacta, mediante la codificación *Run-Level*, donde *run* indica el número de ceros antes del coeficiente no nulo y *level* la magnitud de ese coeficiente. Los coeficientes DCT de frecuencias mayores normalmente se mapean a cero, por lo que los últimos valores de un vector reordenado serán ceros. Para indicar el último coeficiente nulo de la secuencia se añade otro parámetro: *last*, de manera que cada coeficiente se representa finalmente con 3 valores. En el último coeficiente el parámetro *last* se cambia a uno. La tabla 1 muestra un ejemplo:

Tabla 1: Codificación Run-Level

Coeficientes originales	20	0	15	-3	0	0	0	-7
Codificación *Run-Level*	(0,20), (1,15), (0,-3), (3,-7),...							
Codificación *Run-Level-Last*	(0,20,0), (1,15,0), (0,-3,0), (3,-7,1).							

Transformada DWT: usando este tipo de transformada, muchos de los coeficientes en subbandas altas son muy pequeños y al cuantizarlos deben convertirse en ceros sin que se aprecie una pérdida significativa de la calidad de imagen. Los coeficientes no nulos tienen a corresponder con estructuras de la imagen, por ejemplo, con el violín de la ilustración 14. Si un coeficiente de la subbanda de baja frecuencia, LL, es no nulo, hay una gran probabilidad de que los coeficientes en las posiciones correspondientes de las demás subbandas sean también no nulos.

Para mejorar la eficiencia de la codificación se construye un árbol de coeficientes cuantizados no nulos, empezando por la subbanda raíz, la de menor frecuencia. Por ejemplo, si se han usado dos niveles de descomposición, un sólo coeficiente de la banda LL de la capa 1 tiene un coeficiente correspondiente en las otras bandas de la capa 1 (todos corresponden a la misma región de la imagen original). De forma análoga, el coeficiente de la capa 1 tiene otros tres coeficientes correspondientes en la capa 2, coeficientes hijo.

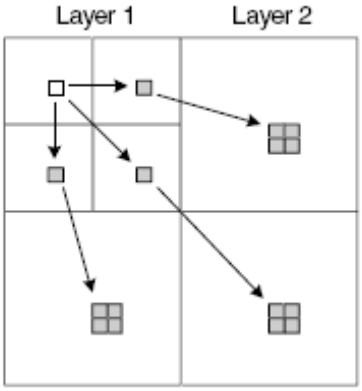

Ilustración 17: Coeficientes de las capas con la DWT

Antes de pasar por el codificador de entropía, se deben representar los datos de la forma más compacta posible. Una forma de conseguirlo es mediante la codificación *zerotree*. Ésta consiste en lo siguiente: se codifica un coeficiente de la primera capa, y se sigue con sus coeficientes hijo. Este proceso se repite hasta que se llega a un coeficiente nulo, cuando es probable que los coeficientes hijo sean también nulos, y entonces se representan todos por un código que identifica el árbol de ceros. El decodificador reconstruye el mapa de coeficientes a partir de la raíz de cada árbol; los coeficientes no nulos se decodifican y reconstruyen, y cuando se encuentra un código de árbol de ceros se ponen a cero los coeficientes hijo. Ésta es la base del método de codificación EZW (Embedded Zero Tree) [3, 11, 18].

2.3.3. Codificador de entropía

El codificador de entropía convierte una serie de símbolos que representan los elementos de una secuencia de vídeo en una secuencia de bits comprimida adecuada para su almacenamiento o transmisión. Los símbolos de entrada pueden contener coeficientes cuantizados de la transformada, vectores de movimiento, cabeceras, marcas de sincronización e información adicional. Esta codificación tiene dos partes: la codificación predictiva, que consiste en explotar la correlación entre regiones del cuadro codificado, y la codificación de entropía en sí, que puede hacerse con dos técnicas: con los códigos modificados de Huffman de longitud variable o con codificación aritmética.

Codificación predictiva: ciertos símbolos están altamente correlacionados en regiones locales de la imagen. Por ejemplo, los coeficientes DC de bloques vecinos codificados de forma *intra* pueden ser bastante parecidos, o los vectores de movimiento de regiones adyacentes pueden tener componentes similares. La efectividad de la codificación puede aumentarse si se predicen los elementos del bloque o macrobloque actual de la información codificada previamente y codificar la diferencia entre la predicción y el valor actual.

La compresión de un vector de movimiento puede mejorarse prediciendo cada vector a partir de otros previamente codificados. Una predicción simple para el vector de movimiento del actual macrobloque X, es el macrobloque adyacente horizontalmente A (en la ilustración 18). También podrían utilizarse para la predicción más vectores codificados previamente (A, B o C en la ilustración). La diferencia entre el vector actual y la predicción (Motion Vector Difference o MVD) se codifica y se transmite.

Ilustración 18: Candidatos para la predicción.

Del mismo modo, en un códec en tiempo real suele ir cambiando ligeramente el paso de cuantización cada cierto número de macrobloques. En lugar de enviar cada vez el nuevo valor que toma éste, se envía la diferencia entre el nuevo y el antiguo, reduciendo el número de bits necesarios para señalizar este cambio.

Codificación de longitud variable: un codificador de longitud variable asigna a los símbolos de entrada una palabra de código de longitud variable, a la que se suele llamar VLC (Variable Length Code). Los símbolos que ocurran frecuentemente se representan con las palabras más cortas mientras que a los menos probables se les asignan las palabras más largas. En una secuencia larga de símbolos esto se traduce en una apreciable compresión de los datos.

Los códigos de Huffman, por ejemplo, se basan en la probabilidad de ocurrencia de un símbolo al asignar las palabras. En el esquema original propuesto es necesario calcular la probabilidad de cada símbolo antes de construir las palabras del código. Este método de codificación presenta dos principales problemas: el decodificador debe tener la tabla de códigos, lo que supone enviar información extra de cabecera y además, para secuencias largas de vídeo, las probabilidades no pueden calcularse hasta que no se haya procesado la secuencia completa, lo que puede introducir retrasos en la codificación. Por estos motivos, se utilizan los que se denominan códigos de Huffman precalculados.

Los códigos de Huffman precalculados se basan en las distribuciones de probabilidad de vídeos genéricos. En el perfil simple de MPEG-4 se utilizan los siguientes códigos precalculados:

- TCOEF (Transform Coefficients): se definen 102 combinaciones posibles de (*run*, *level*, *last*). A cada una le corresponde un código VLC de hasta 13 bits, donde el último representa el signo del coeficiente. El resto de combinaciones se codifican usando la secuencia de escape, seguida de un código de longitud fija de 13 bits que describe los valores de *run*, *level* y *last.*
- MVD (Motion Vector Difference): los vectores de movimiento diferenciales codificados se representan mediante dos códigos VLC: uno para la componente *x* y otro para la componente *y*

Estos códigos se parecen a los de Huffman en que cada símbolo se representa con una palabra única y a los símbolos más comunes se le asignan palabras más cortas. En cambio, se diferencian en que se basan en distribuciones de probabilidad genéricas y sólo definen 102 códigos variables, por lo que el resto tienen que codificarse con códigos de longitud fija.

Codificación aritmética: un codificador aritmético convierte una secuencia de símbolos en un único número fraccional, dentro del rango [0,1], en lugar de utilizar palabras, lo que hace el código más eficiente. La idea es tener una línea de probabilidad [0,1] y asignar a cada símbolo una porción de esta línea. El tamaño de esta porción será directamente proporcional a la frecuencia con que aparece el símbolo en la secuencia. Al codificar un

símbolo, el intervalo actual se divide en subintervalos como en el paso anterior. El nuevo intervalo será la base para la codificación del siguiente símbolo.

La principal ventaja de esta codificación es que el número transmitido no está sujeto a un número fijo de bits para cada símbolo transmitido, por lo que normalmente supera en eficiencia a los códigos de Huffman. La principal desventaja es que se debe recibir la palabra entera para empezar a decodificar y si un solo bit se corrompe, el mensaje entero puede estar mal [3, 6, 18].

2.4. MPEG-4 VISUAL: Perfil Simple

El estándar ISO/IEC 14496-2, MPEG-4 VISUAL, supera al anterior, MPEG-2, tanto en términos de eficiencia de la compresión, más compresión para la misma calidad visual, como en flexibilidad, permitiendo un mayor número de aplicaciones. Esto se ha conseguido de dos maneras: mediante el uso de algoritmos más avanzados de compresión y la proporción de un extenso conjunto de herramientas para la codificación y manipulación de información digital multimedia.

MPEG-4 VISUAL consiste en un núcleo principal -el modelo de codificación híbrido DPCM/DCT- junto con un conjunto de aplicaciones que amplían sus funciones principales, entre las que se encuentran: eficiencia mejorada de compresión, la transmisión fiable, la codificación de objetos dentro de una escena visual o modelos de animación de cuerpo o cara.

El estándar define una serie de perfiles, grupos recomendados de herramientas, de forma que en cada aplicación pueda usarse el perfil más apropiado. Entre estos perfiles se encuentran el *Simple*, con un conjunto pequeño de aplicaciones, el *Core* o *Main*, con herramientas para la codificación de múltiples objetos de formas arbitrarias, el *Advanced Real Time Simple*, con herramientas de tratamiento de errores en transmisiones de bajo retraso, o el *Advanced Simple*, que proporciona una compresión mejorada utilizando algoritmos más complejos [7].

Una de las novedades que introduce MPEG-4 Visual es que sustituye la secuencia tradicional de vídeo, compuesta por una colección de cuadros rectangulares de vídeo, por una secuencia de objetos de vídeo. Un objeto de vídeo se define como: “entidad flexible a la que se puede acceder y manipular” y consiste en un área de forma aleatoria de la escena de vídeo que puede durar un tiempo no definido. Un objeto de vídeo en un instante de tiempo en particular es un VOP (Video Object Plane). La introducción de este concepto se traduce en una mayor flexibilidad en la codificación de vídeo. Los objetos pueden codificarse independientemente y con diferentes calidades visuales y resoluciones, de forma que se refleje su importancia en la imagen final y haciendo así la compresión más eficiente.

A pesar de la potencial flexibilidad de la codificación basada en objetos, la aplicación de MPEG-4 Visual más extendida es la codificación de cuadros completos de vídeo. Las herramientas necesarias para manejar VOPs rectangulares, normalmente cuadros completos, se agrupan en los perfiles simples. Las herramientas básicas de éstos son similares a los estándares anteriores, la codificación de macrobloques basada en DCT con predicción de compensación de movimiento.

El Perfil Simple se basa en el modelo híbrido de codificación DPCM/DCT con herramientas adicionales para mejorar la eficiencia de la codificación y la transmisión. El formato de la secuencia de vídeo de entrada a un codificador de MPEG-4 es de 4:2:0, 4:2:2 o 4:4:4, progresiva o entrelazada.

Un codificador compatible con el Perfil Simple debe ser capaz de codificar y decodificar objetos de vídeo simples usando las siguientes herramientas:

- I-VOP: codificación de VOPs rectangulares en modo *intra*, con formato de vídeo progresivo.
- P-VOP: codificación de VOPs rectangulares en modo *inter*, con formato de vídeo progresivo.
- Modo de cabecera corta (para la compatibilidad con otro estándar).
- Herramientas para la eficiencia de la compresión: cuatro vectores de movimiento por macrobloque, vectores de movimiento sin restricciones, predicción *intra*.
- Herramientas de eficiencia de transmisión: paquetes de vídeo, partición de datos, códigos de longitud variable reversibles, etc.

2.4.1. Herramientas básicas de codificación

2.4.1.1. I-VOP

Un I-VOP rectangular es un cuadro de vídeo codificado en modo *intra*, es decir, sin predicciones a partir de otro VOP codificado. Los procesos de codificación y decodificación se muestran en la ilustración 19:

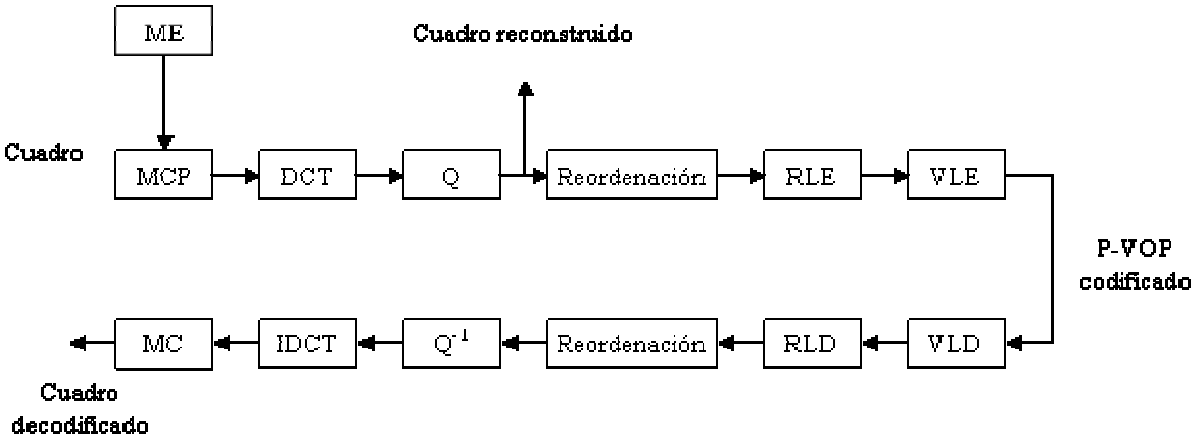

Ilustración 19: Codificación y decodificación de un I-VOP.

- DCT e IDCT: los bloques de muestras de luminancia y croma, de tamaño 8x8, se transforman usando la DCT hacia delante en la codificación y la DCT inversa en la decodificación.
- Cuantización: el estándar MPEG-4 Visual especifica el método de reescalado. Éste se controla con el paso de cuantización, *QP*, que puede tomar valores en el intervalo [1,31]. Se describen dos métodos de reescalado en el estándar: el *método 2* o básico y el *método 1*, que usa una matriz de cuantización, siendo más flexible pero más complejo. El *método 2* hace la cuantización inversa de la siguiente manera: el coeficiente DC en un macrobloque intracodificado se reescala así:

$$DC = DC_Q \cdot dc\_scaler$$

donde DC_Q es el coeficiente cuantizado, *DC* es el coeficiente reescalado y *dc_scaler* es un parámetro definido en el estándar. En el modo de cabecera corta, *dc_scaler* es 8, en el resto de los casos se calcula en función del parámetro *QP*. El resto de coeficientes, *AC* y *DC* tipo *inter*, se reescalan de esta forma:

$$|F| = QP \cdot \left(2 \cdot |F_Q| + 1\right) \quad si\ QP\ es\ impar\ yF_Q \neq 0$$

$$|F| = QP \cdot \left(2 \cdot |F_Q| + 1\right) - 1 \quad si\ QP\ es\ par\ yF_Q \neq 0$$

$$F = 0 \quad si\ F_Q = 0$$

donde F_Q es el coeficiente cuantizado y F el coeficiente reescalado. El signo de F es el mismo que el de F_Q. La cuantización hacia delante no se define en el estándar.

- Reordenación en zig-zag: los coeficientes DCT se reordenan en zig-zag antes de codificarlos.
- Codificación *Run-Level* (RLE, Run Level Encoding): el vector de coeficientes cuantizados correspondientes a cada bloque se codifica para representar los coeficientes nulos de forma eficiente. Cada coeficiente no nulo se representa con el trío (*run*, *level*, *last*), donde *last* indica si es el último coeficiente, *run* indica el número de ceros antes del coeficiente no nulo y *level* indica el valor y signo del coeficiente.
- Codificación de entropía o codificación con VLCs (VLE): la información de cabecera y los tríos (*run*, *level*, *last*), se representan con códigos de longitud variable (VLC). Estos códigos son similares a los códigos de Huffman pero están basados en probabilidades precalculadas y están definidos en el estándar.

Un I-VOP codificado consiste en una cabecera, otras cabeceras opcionales y los macrobloques codificados. Cada macrobloque se codifica con una cabecera (que define el tipo de macrobloque, indica cuáles de los bloques contienen coeficientes codificados, señaliza los cambios en el parámetro de cuantización, etc.) seguida de los coeficientes codificados de cada bloque 8x8.

En el decodificador, la secuencia de códigos VLC se decodifica para extraer los coeficientes de transformada cuantizados, que se reescalan y se transforman con la IDCT para reconstruir el I-VOP decodificado.

2.4.1.2. P-VOP

Un P-VOP se codifica con predicción *intra* a partir de un cuadro previamente codificado de tipo I o P. Los procesos de codificación y decodificación se muestran en la ilustración 20:

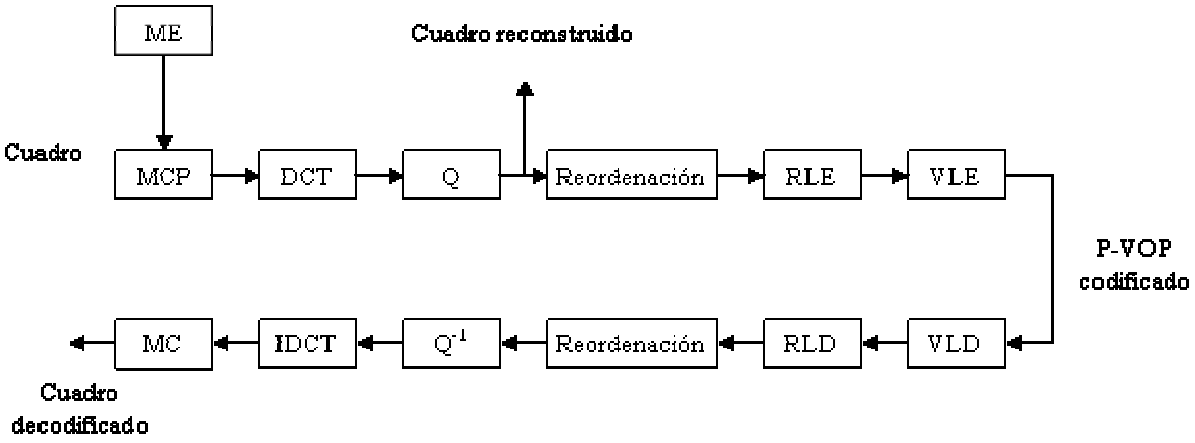

Ilustración 20: Codificación y decodificación de un P-VOP.

Estimación (ME) y compensación de movimiento (MCP): la compensación de movimiento se basa en macrobloques de 16x16 píxeles. La diferencia entre el macrobloque actual y la región de compensación en el cuadro de referencia, el vector de movimiento, puede tener resolución de una porción de píxel. En este caso se realiza una interpolación para obtener muestras en estas posiciones. El método de la estimación de movimiento se deja al diseñador. La región que más se parece se resta al macrobloque actual para producir un macrobloque residual.

Tras la compensación de movimiento, el macrobloque residual pasa por el resto de los procesos: transformada DCT, cuantización, reordenación, codificación *Run-Level* y codificación entrópica. El residual cuantizado se reescala y pasa por la transformación inversa para la reconstrucción de una copia local del macrobloque decodificado.

Un P-VOP codificado consiste en una cabecera de VOP, cabeceras opcionales, macrobloques codificados con sus cabeceras (incluyendo esta vez los vectores de movimiento diferenciales codificados) y coeficientes residuales codificados de cada bloque 8x8.

El decodificador forma la misma predicción de la compensación de movimiento a partir del vector de movimiento y de su copia local del VOP de referencia. La información residual decodificada se añade a la predicción para reconstruir el macrobloque decodificado.

Los macrobloques en un P-VOP pueden estar codificados en modo *inter* (con predicción de compensación de movimiento) o en modo *intra* (sin predicción de movimiento). El modo *inter* dará más eficiencia en la codificación pero el modo *intra* será útil en regiones donde no haya parecido con regiones de cuadros anteriores.

2.4.1.3. Modo de cabecera corta

La herramienta de cabecera corta proporciona compatibilidad entre MPEG-4 Visual y el estándar de la ITU-T H.263. En este modo, los macrobloques de un VOP se organizan en grupos de bloques (Group Of Blocks, o GOBs), consistiendo cada uno en una o más filas completas de macrobloques y separados, opcionalmente, por marcas de sincronización.

2.4.2. Herramientas para la eficiencia en la codificación

Existen una serie de herramientas en el Perfil Simple de MPEG-4 que mejoran la eficiencia en la codificación. Éstas se utilizan cuando el modo de cabecera corta no está en uso.

2.4.2.1. Cuatro vectores de movimiento por macrobloque

La compensación de movimiento tiende a ser más efectiva con tamaños de bloque más pequeños. El tamaño de bloque por defecto de la compensación de movimiento es 16x16 muestras de luminancia y 8x8 muestras de croma, con un vector de movimiento por macrobloque. Esta herramienta proporciona al codificador la posibilidad de elegir un menor tamaño de bloque: 8x8 muestras de luminancia y 4x4 de croma, creando cuatro vectores de movimiento por macrobloque. Al aumentar el número de vectores por macrobloque se disminuye la energía del residual que queda al compensar el movimiento, particularmente en áreas de movimiento más complejas como los bordes o en objetos en movimiento. Al mismo tiempo, se incrementa la información de cabecera al mandar cuatro vectores en vez de uno, por lo que el codificador debe elegir el número de vectores en cada macrobloque, según las características, de éste de forma óptima.

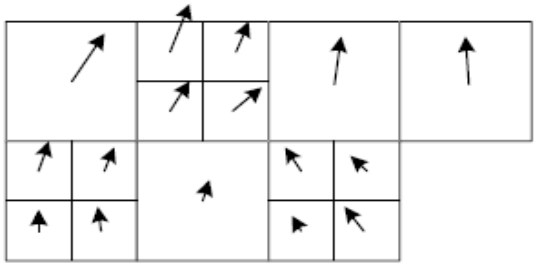

Ilustración 21: Vectores de movimiento en cada macrobloque.

2.4.2.2. Vectores de movimiento sin restricciones

En algunos casos, la región que mejor encaja para un macrobloque de 16x16 corresponde a una fuera de las fronteras del cuadro de referencia. El uso de vectores de movimiento sin restricciones permite a los vectores apuntar a regiones fuera del cuadro de referencia. Si el vector de movimiento está fuera del cuadro, se utiliza la última muestra del borde más cercana. Esta herramienta puede mejorar la eficiencia en la compensación de movimiento, especialmente cuando hay objetos moviéndose de dentro a fuera de la imagen.

2.4.2.3. Predicción modo *intra*

Los coeficientes de transformada de baja frecuencia de bloques 8x8 codificados en modo *intra*, están normalmente correlacionados. De esta forma, el coeficiente DC y, opcionalmente, la primera fila y columna de coeficientes AC de un bloque 8x8 se predicen a partir de bloques vecinos.

El coeficiente DC del bloque actual (X) se predice del coeficiente DC del bloque superior, C, o del bloque izquierdo, A, previamente codificados. Los valores de los coeficientes DC reescalados de los bloques A, B y C determinan el método de predicción para el coeficiente DC. Si A, B o C están fuera de las fronteras del VOP, o no están codificados en modo *intra*, el valor del coeficiente correspondiente se fija en 1024 ($2^{(\text{bits x píxel}+2)}$), valor que corresponde al coeficiente DC de un bloque gris de tono medio. La dirección de la predicción se determina así:

Tabla 2: Predicción de coeficientes DC.

Condición	Bloque para la predicción
$\lvert DC_A - DC_B \rvert < \lvert DC_B - DC_C \rvert$	C
$\lvert DC_A - DC_B \rvert >= \lvert DC_B - DC_C \rvert$	A

La dirección del menor gradiente del coeficiente DC se elige como la dirección de predicción del bloque X. La predicción, PDC, se forma dividiendo el coeficiente DC elegido por un factor de escalado, y PDC se resta del actual coeficiente cuantizado para obtener el residual, que se codifica y se transmite.

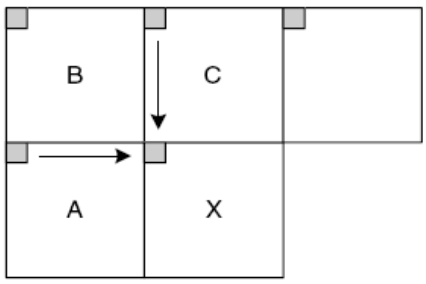

Ilustración 22: Predicción de los coeficientes DC

La predicción de los coeficientes AC se hace de forma parecida, con la primera fila o columna de coeficientes AC predichos en la dirección marcada por el gradiente del coeficiente DC. Por ejemplo, si la dirección de predicción es del bloque A, la primera columna de coeficientes AC del bloque X se predice de la primera columna del bloque A. Si la dirección de la predicción es del bloque C, la primera fila de coeficientes AC en X se predicen

de la primera fila de C. La predicción se reescala dependiendo del paso de cuantización utilizado para los bloques X, A o C.

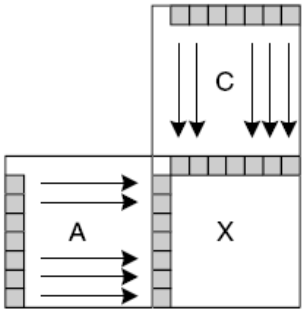

Ilustración 23: Predicción de los coeficientes AC

Este proceso se repite independientemente para cada bloque del macrobloque usando el apropiado bloque adyacente para la predicción. Las predicciones se forman de igual manera para la luminancia y cada una de las componentes de la croma.

2.4.3. Herramientas para la eficiencia en la transmisión

Un error en un bit o una pérdida de algún paquete puede causar en el decodificador una pérdida de la sincronización con la secuencia de códigos VLC. Esto puede causar que se decodifiquen de forma errónea algunos o todos los datos que siguen al error, lo que se traducirá en que parte o todo el VOP se verá distorsionado, ya que el error se propaga de forma espacial. Si los VOPs siguientes se predicen a partir del VOP dañado, el error también se propagará de forma temporal.

La propagación de errores puede evitarse insertando en la secuencia de bits un código binario único que identifique un punto de sincronización. Si el decodificador detecta un error, un VLC erróneo por ejemplo, aplica un mecanismo de recuperación que consiste en buscar el siguiente punto de sincronización.

Existen varias herramientas que se han diseñado para mejorar la calidad de la transmisión de los datos codificados, que sólo se utilizan cuando el modo de cabecera corta no está activo.

2.4.3.1. Empaquetamiento de vídeo

Un VOP transmitido consiste en uno o varios paquetes de vídeo. Un paquete de vídeo consta de una marca de sincronización, un campo de cabecera y una serie de macrobloques codificados. A la marca de sincronización le sigue el número de macrobloque, lo que permite al decodificador posicionarse en el primer bloque del paquete correctamente. Después, se define el parámetro de cuantización y una bandera, HEC (Header Extension Code). Si HEC

es '1', le sigue un duplicado de la cabecera del VOP, se aumenta la información que se transmite pero permite al decodificador recuperar la cabecera si en la primera ha habido errores.

Esta herramienta ayuda a la recuperación frente a los errores de dos maneras: las marcas de sincronización evitan que el error se propague entre distintos paquetes de vídeo y el parámetro HEC permite recuperar una cabecera de VOP que se haya perdido de otro punto dentro del VOP.

2.4.3.2. Partición de datos

La partición de datos consiste en que el codificador reordene los datos de forma que la propagación de errores de transmisión se haga más difícil. El paquete de vídeo se divide en dos partes. La primera, después de la cabecera, contiene la información del modo de codificación de cada macrobloque junto con los coeficientes DC de cada bloque (para macrobloques *intra*) o vectores de movimiento (para macrobloques *inter*). El resto de los datos (coeficientes AC y coeficientes DC para macrobloques *inter*) se colocan en la segunda partición de datos después de la marca de sincronización. La información de la primera partición se considera más importante para la correcta decodificación del paquete de vídeo. Si la primera partición se recupera, es posible reconstruir el resto del paquete de vídeo, incluso si la segunda partición se ha dañado o perdido.

2.4.3.3. Códigos VLC reversibles

Un conjunto opcional de códigos VLC reversibles (RVLCs) se pueden usar para codificar los coeficientes DCT. Estos códigos tienen la particularidad de poder ser correctamente decodificados en las dos direcciones: hacia delante y hacia atrás; esto hace posible que se minimice el área de la imagen afectada por el error.

El decodificador primero decodifica los datos hacia delante y, si detecta un error, el paquete se decodifica hacia atrás desde la siguiente marca de sincronización. El uso de esta herramienta hace que el daño causado por el error pueda reducirse a sólo un macrobloque.

2.5. Estructura de la secuencia de vídeo codificada

En general la secuencia de datos codificados se puede ver como una jerarquía en la que cada estructura sintáctica contiene una o más estructuras subordinadas.

2.5.1. Visual Object Sequence

Visual Object Sequence es la estructura sintáctica más alta de la secuencia de vídeo codificada. Ésta comienza con el código *visual_object_sequence_start_code* y le sigue uno o

más objetos *visual objects* codificados. La estructura termina con el código *visual_object_sequence_end_code.*

2.5.2. Visual Object

Un *Visual Object* comienza con el código *visual_object_start_code*, le siguen los parámetros que indican el perfil y el nivel del vídeo y una identificación del objeto visual. Tras estos parámetros comienza un *video object*.

2.5.3. Video Object

Un *Video Object* comienza con el código *video_object_start_code* y le siguen una o más *video object layers*.

2.5.4. Video Object Layer

La capa *Video Object Layer* comienza con el código *video object_layer_start_code* y le sigue una cabecera que describe parámetros, entre los que se encuentran:

- Tipo de objeto de vídeo: *Simple*, *Advanced*, *Core*, etc.
- Relación de aspecto: 4:3, 16:9, etc.
- Formato de croma (4:4:4, 4:2:2, 4:2:0).
- Forma de la *Video Object Layer*: rectangular, binary, binary only, grayscale.
- Muestreo del vídeo: progresivo o entrelazado.

Tras la cabecera siguen una serie de *Video Object Planes* o VOPs.

2.5.5. Video Object Plane

El *Video Object Plane* comienza con el código *vop_start_code* y le siguen una serie de parámetros de éste, entre los que se encuentran:

- Tipo de VOP: I o P.
- Vop_coded: indica si el VOP está codificado o no, en el caso de que no lo esté ya no hay más información sobre éste.
- Dimensiones del VOP, en píxeles.
- Paso de cuantización.

Una vez se terminan los parámetros, comienzan los datos de los macrobloques. Según el tipo de VOP, los macrobloques que tendrá podrán ser de un tipo u otro. Si el VOP es de tipo I, entonces sólo podrá tener macrobloques tipo *intra*, mientras que si el VOP es de tipo P, podrán ser *intra* o *inter*.

2.5.6. Macrobloque

La capa de macrobloque tiene la siguiente estructura:

coded	mcbpc	ac_pred	cbpy	dquant	mvd	mvd_{2-4}	bloques

Ilustración 24: Estructura de un macrobloque

- **Coded**: cada macrobloque comienza con un parámetro que indica en primer lugar si el macrobloque está codificado o no. En los VOPs de tipo I, todos los macrobloques están codificados, pero en los de tipo P, pueden estarlo o no. Si el macrobloque no está codificado ya no se envía más información sobre él y el decodificador trata el macrobloque como tipo *inter*, con vector de movimiento nulo y sin coeficientes DC.
- **Mcbpc**: esta secuencia de bits describe el tipo de macrobloque y el patrón de la codificación de bloques de croma. La tabla 3 resume los tipos de macrobloque para cada tipo de VOP:

Tabla 3: Tipos de macrobloque

Tipo VOP	MB type	Nombre
P	Not coded	-
P	0	inter
P	1	inter+q
P	2	inter4v
P	3	intra
P	4	intra+q
P	stuffing	
I	3	intra
I	4	intra+q

- **Ac_prd**: es una bandera que si está a 1 indica que o la primera fila o la primera columna de coeficientes AC están diferencialmente codificados para los macrobloques de tipo *intra*.
- **Cbpy**: es un código de longitud variable que representa el patrón de codificación de los bloques de luminancia no transparentes con al menos un coeficiente DC en un macrobloque.
- **Dquant**: es un código de 2 bits que indica el cambio en el parámetro de cuantización.
- **Mvd**: son dos códigos VLC, uno para el vector de movimiento diferencial horizontal y otro para el vertical.

- **Mvd $_{2\text{-}4}$**: si el macrobloque es de tipo 2 entonces el macrobloque tiene 4 vectores de movimiento, por lo que se añaden estos tres, cada uno con sus componentes horizontal y vertical.

2.5.7. Bloques

Un macrobloque contiene 6 bloques: cuatro de luminancia y dos de croma. El patrón de codificación de luminancia y croma descritos en la cabecera del macrobloque indicarán si cada bloque está codificado o no. Si no está codificado se pasa al siguiente bloque.

Cada bloque codificado comienza con un código que define el tamaño del coeficiente DC en bits. A éste le sigue el coeficiente DC diferencial, de croma o luminancia según el bloque que sea.

Después del coeficiente DC se encuentran una serie de códigos VLC que representan los tríos (*run, level, last*) correspondientes a los coeficientes AC del bloque. El siguiente bloque comienza tras el coeficiente que tiene *last*.

La reconstrucción de los VOPs codificados comenzará con la obtención del vector de movimiento del macrobloque añadiendo la predicción al parámetro MVD incluido en la secuencia de bits. Tras esto, se procederá al reescalado, reordenación y la transformación inversa de los coeficientes. Al compensar el movimiento y decodificar los coeficientes, se realiza una reconstrucción de cada bloque de luminancia y croma.