Departamento de Informática
Universidad de Almería

Transmisión Progresiva de Vídeo JPEG2000 con Movimiento Compensado

Autor
José Carmelo Maturana Espinosa

Director
Dr. Vicente González Ruiz

Tesis Doctoral — Febrero, 2020

José Carmelo Maturana Espinosa Transmisión Progresiva de Vídeo JPEG2000 con Movimiento Compensado 2020

UNIVERSIDAD DE ALMERÍA

TRANSMISIÓN PROGRESIVA DE VIDEO JPEG2000 CON MOVIMIENTO COMPENSADO

Autor
JOSÉ CARMELO MATURANA ESPINOSA

Director
Dr. VICENTE GONZÁLEZ RUIZ

Departamento de Informática
Almería, 2020

UNIVERSITY OF ALMERÍA

PROGRESSIVE TRANSMISSION OF MOTION-COMPENSATED JPEG2000 VIDEO

Author
JOSÉ CARMELO MATURANA ESPINOSA
Supervisor
Dr. VICENTE GONZÁLEZ RUIZ

Department of Informatics
Almería, 2020

Título en Español: *Transmisión Progresiva de Vídeo JPEG2000 con Movimiento Compensado.*
Título en Inglés: *Progressive Transmission of Motion-Compensated JPEG2000 Video.*
Programa de doctorado: 8908 Doctorado en Informática (RD99/11).
Doctorando: José Carmelo Maturana Espinosa.
Director: Vicente González Ruiz.

Texto impreso en Almería, 10 de febrero de 2020

” If I have seen further, it is by standing on the shoulders of giants.”

- Isaac Newton

Resumen

El objetivo principal de esta tesis ha sido crear un códec de vídeo denominado MCJ2K (Motion Compensated JPEG2000) con dos características esenciales: (1) compatible, sin necesidad de alterar ninguno de los servidores JPIP (Protocolo Interactivo JPEG2000) actuales y (2) capaz de generar un flujo de información con una relación R/D (rate-distorsión) competitiva. Para lograr un alto rendimiento R/D, nuestra propuesta MCJ2K: (1) genera un code-stream escalable con múltiples puntos de truncado, adaptando así la transmisión a las necesidades del cliente, y (2) explota la correlación temporal presente en la mayoría de las secuencias de imágenes, a través de la compensación de movimiento.

El code-stream producido por MCJ2K puede ser distribuido por servidores JPIP estándar, gracias al uso de formatos de archivo estándar J2K. En escenarios con limitaciones de ancho de banda, una cuestión importante en MCJ2K es determinar la cantidad de datos de cada subbanda temporal que debe transmitirse para maximizar la calidad de las reconstrucciones en el lado del cliente. Para resolver este problema, hemos propuesto dos algoritmos de rate-allocation que proporcionan reconstrucciones de calidad progresiva. El primero, OSLA (Optimized subband Layer Allocation, en español asignación optimizada de capas de subbandas), determina la mejor progresión (en calidad) capa a capa, aunque es computacionalmente costoso. El segundo, ESLA (Estimated-slope subband Layer Allocation, en español asignación estimada de capas de subbandas), es sub-óptimo en la mayoría de los casos, pero muy rápido y más conveniente en escenarios de transmisión en tiempo real.

Una comparación experimental muestra que, incluso cuando se utiliza un esquema de compensación de movimiento básico, no sólo el rendimiento R/D de MCJ2K es competitivo en comparación con MJ2K, sino también con respecto a otros códecs de vídeo estándar.

Abstract

The main objective of this thesis was to create a video codec: (1) compatible, without altering any of the current JPIP (JPEG2000 Interactive Protocol) servers and (2) capable of generating a flow of information with a competitive R/D (rate-distortion) ratio. To achieve high R/D performance, our proposal, MCJ2K (Motion Compensated JPEG2000): (1) generates a scalable code-stream, with multiple truncation points, thus adapting the transmission to the client's needs, and (2) exploits the temporal correlation present in most image sequences, through motion compensation.

MCJ2K code-streams can be served by standard JPIP servers, thanks to the use of only J2K standard file formats. In bandwidth-constrained scenarios, an important issue in MCJ2K is determining the amount of data of each temporal subband that must be transmitted to maximize the quality of the reconstructions at the client-side. To solve this problem, we have proposed two rate-allocation algorithms that provide reconstructions that are progressive in quality. The first, OSLA (Optimized subband-Layers Allocation), determines the best progression (quality) layer by layer, but is computationally expensive. The second, ESLA (Estimated-Slope subband-Layers Allocation), is suboptimal in most cases, but much faster and more convenient for real-time streaming scenarios.

An experimental comparison shows that, even when a basic motion compensation scheme used, not only is the R/D performance of MCJ2K competitive compared to MJ2K but also concerning other standard video codecs.

Contexto y marco teórico

Actualmente existen multitud de dispositivos que reciben un streaming bajo condiciones muy diversas de hardware o ancho de banda. Para una buena experiencia de usuario se necesita un códec de vídeo que realice compresiones eficientes en términos de *tradeoff rate-distorsión*. Por un lado, la compensación de movimiento reduce la redundancia de la codificación y mejora (normalmente) el rate-distorsión del vídeo. Por otro lado, el orden en que se transmite un codestream truncado influye en su desempeño rate-distorsión, por lo que su estudio mejorará las transmisiones de vídeo en entornos heterogéneos de streaming de vídeo.

Agradecimientos

A familia y amigos, por su apoyo constante e incondicional.

A Vicente, director de la tesis, por su saber hacer ante todo desafío que la misma nos ha planteado. Desde un principio le agradecí que confiara en mí para continuar su propio trabajo de doctorado. Ahora puedo decir, que me siento orgulloso de tenerle como amigo.

Al grupo Supercomputación-Algoritmos (TIC 146) de la Universidad de Almería, por ofrecerme la oportunidad de formar parte de un equipo de tal valía y poder desarrollar en él esta tesis.

10 de febrero de 2020

Acrónimos

AIA	Atmospheric Imaging Assembly
AVC	Advanced Video Coding
CDF	Cohen–Daubechies–Feauveau
CTU	Coding Tree Units
DASH	Dynamic adaptive streaming over HTTP
dB	Decibel
DCT	Discrete Cosine Transform
DVB	Digital Video Broadcasting
DWT	Discrete Wavelet Transform
DZQ	Deadzone Scalar Quantization
EBCOT	Embedded Block Coding with Optimal Truncation
ESLA	Estimated-Slope subband-Layers Allocation
FAST	Rate Allocation Through Steepest Descent
FEC	Forward Error Correction
GOP	Group Of Pictures
H	High frequency subband
HEVC	High Efficiency Video Coding
ICT	Irreversible Color Transformation
IEC	International Electrotechnical Commission
IPTV	Internet Protocol Television
ISO	International Organization for Standardization
ITU	International Telecommunication Union
J2K	JPEG2000
JPIP	J2K Interactive Protocol
Kbps	Kilobits per second
L	Low frequency subband
LIMAT	Lifting-based Invertible Motion Adaptive Transform
LRCP	Quality Layers, Resolution, Components and Precincts
MC	Motion Compensated
MCJ2K	Motion Compensated JPEG2000
MCTF	Motion Compensated Temporal Filtering
MDC	Multiple Description video Coding
ME	Motion Estimation
MJ2K	Motion JPEG2000
MPEG	Moving Picture Experts Group
MV	Motion Vector
OSLA	Optimized subband-Layers Allocation
P	Coding Pass
P2P	Peer-to-Peer
PCRD-opt	Post-Compression Rate/Distortion optimization
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
R/D	Rate-Distortion
RA	Rate-Allocation

ACRÓNIMOS

RDO	Rate-Distortion Optimization
RGB	Red, Green, Blue
RLCP	Resolution, Quality Layers, Components and Precincts
ROI	Region of Interest
RSVP	Resource Reservation Protocol
RTC	Reversible Color Transformation
SDO	Solar Dynamics Observatory
SDTV	Standard Definition TV
SHVC	Scalable High Efficiency Video Coding
SL	Subband-Layer
SNR	Signal-to-Noise Ratio
SVC	Scalable Video Coding
TRL	Temporal Resolution Level
UHDTV	Ultra-High Definition TV
VLC	Variable Length Coding
VoD	Video on Demand
YCbCr	Luminance & Chrominance
YUV	Intensity, Hue and Value

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo

1

Introducción

Este capítulo presenta los fundamentos de codificación de imagen y vídeo que son necesarios para describir nuestra propuesta, MCJ2K (Motion Compensated JPEG2000), un códec de vídeo escalable con compensación de movimiento y técnicas de rate-allocation. En la Sec. 1.1 se exponen las posibilidades que la escalabilidad en imagen y vídeo proveen. En la Sec. 1.2 se describe el compresor de imagen JPEG2000, que es la base para presentar en la Sec. 1.3 Motion JPEG2000, un códec de vídeo simple en el que nos basamos para crear nuestra propuesta. Además se exponen las principales técnicas de codificación de vídeo: (1) la compensación de movimiento y (2) técnicas de rate-allocation. Finalmente, en la Sec. 1.4 se presentan las ventajas de la transmisión de vídeo escalable sobre la no escalable.

1.1 Escalabilidad de imagen y vídeo

Una imagen o vídeo se dice escalable si tienen algún tipo de flexibilidad a la hora de descodificar el code-stream. Dependiendo del grado de esta flexibilidad, se pueden obtener 4 tipos de escalabilidades.

1.1.1 Escalabilidad espacial

La escalabilidad espacial se refiere a la posibilidad de descomprimir sólo una versión espacial reducida de las imágenes del code-stream (véase la Fig. 1.1). Definimos un nivel de resolución espacial p como

$$V^{<p>} = \{\frac{Y}{2^p} \times \frac{X}{2^p} \text{ versión de } V_i;\ 0 \leq i < \#V\}, \tag{1.1}$$

Figura 1.1: Escalabilidad espacial de un vídeo.

donde X e Y son las dimensiones de las imágenes. Nótese que $V = V^{<0>}$, donde V es la señal original.

1.1.2 Escalabilidad en calidad

La escalabilidad en calidad se refiere a la posibilidad de descomprimir una versión de calidad reducida de las imágenes del code-stream (véase la Fig. 1.2). Definimos un nivel de calidad q como

$$V^{[q]} = \{\mathcal{Q}^q(V_i);\ 0 \leq i < \#V\}, \tag{1.2}$$

donde $\mathcal{Q}$ es un cuantificador (véase la Fig. 1.8) aplicado típicamente en el dominio de la transformada, con el objetivo de eliminar la información visual menos importante, y q es el paso de cuantificación (en inglés, quantization step). También definimos que $V = V^{[1]}$.

1.1.3 Ventana de interés

En el streaming interactivo de imágenes los usuarios recuperan *versiones* de las imágenes, que se adaptan al hardware y ancho de banda del cliente. Estas versiones llamadas WOI (Windows of Interest, en español región o ventana de interés) es una región (rectangular o cuadrada) definida en tiempo de descompresión.

Figura 1.2: Escalabilidad en calidad de un vídeo.

Esto es especialmente útil en el streaming de imagen o vídeo de alta resolución espacial o profundidad de bits. Téngase en cuenta que una WOI es una región cuyo tamaño y posición puede ser controlado por el usuario en cualquier momento de la transmisión. Su finalidad consiste en poder solicitar una calidad y/o resolución determinadas de esas regiones de la imagen. Esta transmisión selectiva de una región de las imágenes reduce significativamente el consumo de ancho de banda, haciendo un uso eficiente del mismo.

La Fig. 1.3 muestra un ejemplo de interacción con el servicio JHelioviewer [38, 39]. Las imágenes del Sol pertenecen a la agencia AIA (Atmospheric Imaging Assembly). Supongamos que un usuario con una resolución de pantalla de 2048×2048 o un ancho de banda que no permite el streaming a mayor resolución se encuentra recibiendo un vídeo en una versión de baja resolución del mismo (sub-figura izquierda). A continuación solicita una WOI, que en este ejemplo comienza en el píxel 1000×500 de dicha versión a baja resolución de la imagen y tiene un tamaño en píxeles de 1024×1024 (sub-figura central). A partir de aquí el usuario recibe esta WOI en una resolución mayor (sub-figura derecha), es decir, sólo se transmitirá el code-stream relacionado con esa WOI.

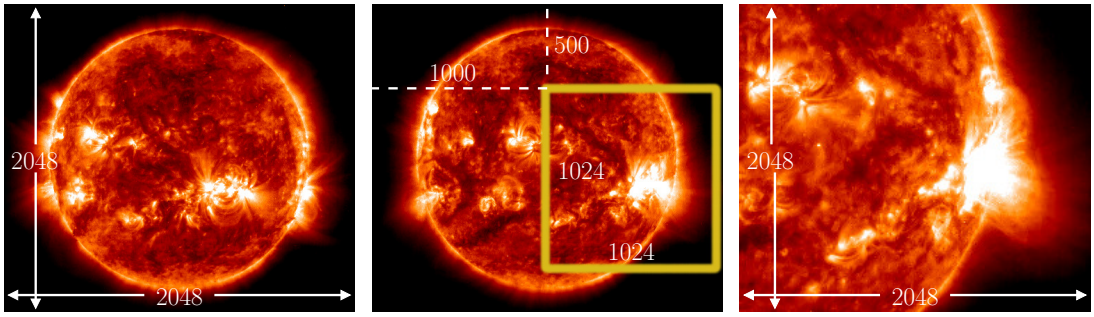


Figura 1.3: Ejemplo de interacción con el servicio JHelioviewer. Crédito de la imagen: NASA/SDO/AIA.

1.1.4 Escalabilidad temporal

La escalabilidad temporal afecta a la codificación y descodificación del vídeo. Se refiere a la posibilidad de reconstruir un conjunto de imágenes equidistantes temporalmente entre sí, de forma que se pueda descodificar el vídeo a un frame/rate o imágenes por segundo determinado. Para esta reconstrucción a menor número de imágenes/segundo deben usarse sólo los datos del code-stream relacionados con las imágenes que finalmente serán mostradas. Según las técnicas usadas en la codificación del vídeo existen códecs de vídeo escalables temporalmente con distinto coste para la escalabilidad temporal.

1.2 Codificación de imagen escalable

La compresión de una imagen consiste en reducir los datos redundantes o irrelevantes de la imagen, con la menor pérdida de información posible. Existen dos tipos de codificadores que comprimen las imágenes. Por un lado, los codificadores sin pérdida, permiten que a partir del code-stream sea posible reconstruir el vídeo original sin ninguna distorsión. Por otro lado, los codificadores con pérdida, sólo permiten una reconstrucción aproximada del original a cambio de conseguir una mayor compresión.

La codificación escalable de imágenes permite descomprimir sólo la parte de la información del code-stream que satisfaga las necesidades del cliente, optimizando por ejemplo, el uso del ancho de banda.

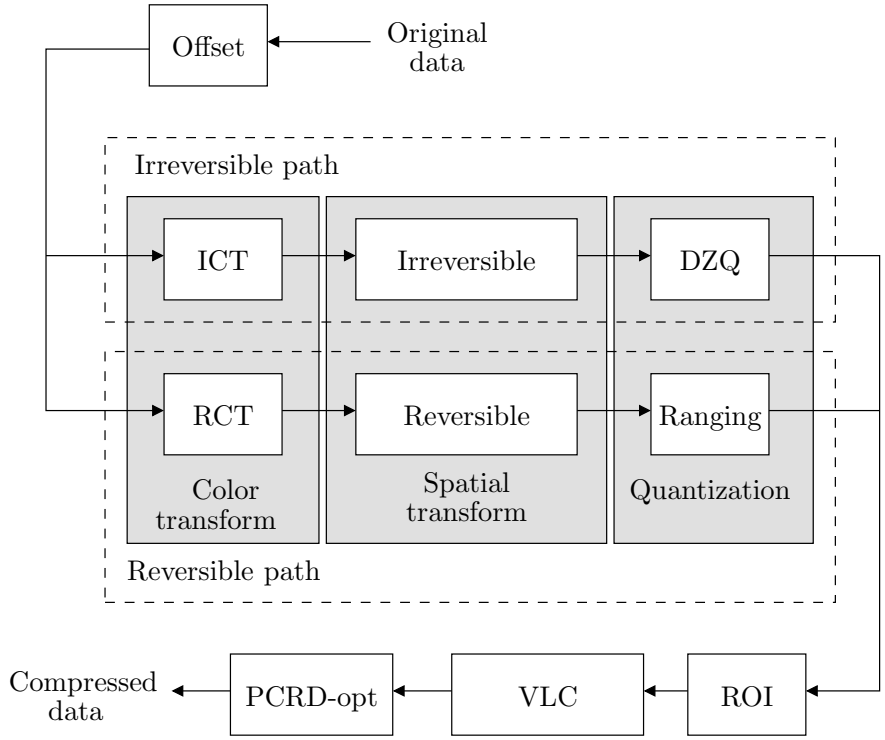


Figura 1.4: Arquitectura del compresor JPEG2000.

1.2.1 **JPEG2000**

JPEG2000 (J2K) [11, 21] es el último estándar de compresión de imágenes desarrollado por el ISO/ITU (International Telecommunication Union). El objetivo de esta sección. es proporcionar una breve descripción a alto nivel del códec JPEG2000 que permita comprender las siguientes secciones de este trabajo. La Fig. 1.4 muestra un diagrama de bloques del codificador JPEG2000. Se trata de un códec de imagen basado en las siguientes etapas secuenciales que son descritas a continuación.

Level Offset

Este primer paso, llamado Level Offset o desplazamiento de nivel, normaliza las muestras de la señal entre

$$-2^{B-1} \leq x[n] < 2^{B-1}, \tag{1.3}$$

donde B es la profundidad de bits. Este desplazamiento de las muestras tiene la utilidad de proporcionar una distribución uniforme en torno a cero y no simétricamente sobre cero. Esto será útil cuando en una etapa posterior se aplique el filtrado de paso alto a las muestras de las subbandas producidas por la transformada DWT (Discrete Wavelet Transform, Transformada Discreta Wavelet en español), representada en la figura por la fase Spatial transform.

La motivación para llevar a cabo el offset es que casi todas las muestras de subbandas producidas por la DWT implican un filtrado de paso alto y, por lo tanto, tienen una distribución simétrica sobre cero. Sin la etapa offset, la subbanda LL presentaría una excepción, introduciendo cierta irregularidad en las implementaciones eficientes del estándar. Además con esta transformación las componentes RGB (Red, Green, Blue) al convertirlas en otras componentes con un modelo de color diferente, se reduce la redundancia que existe entre los canales R, G y B, aumentando la tasa de compresión.

Los siguientes pasos consisten en las transformaciones de color, espacial y en la cuantificación de los datos. Cada una de estas etapas puede llevarse a cabo con o sin pérdida de datos. Estos pasos se describen a continuación.

Transformación de color (RTC)

Esta transformación reduce la correlación entre los componentes de color. Es opcional y sólo se puede utilizar cuando los datos de color se expresan en 3 o más componentes y cuando al menos, las tres primeras componentes tienen el mismo peso o relevancia entre ellas y la misma profundidad de color. La transformación suele convertir de RGB (tres colores) a YCbCr. Y representa el componente de luma y las señales Cb y Cr son los componentes de crominancia. Estos componentes de color suelen suponer menos del 20 % [61] de los bits utilizados para representar una imagen a color respecto de la luma.

Existen dos alternativas para transformar el color:

1. *ICT* (Transformación de color irreversible): En dicha transformación de color los componentes se representan en el dominio de punto flotante. La pérdida de precisión al perder decimales provoca una pérdida de datos.

2. *RCT* (Transformación de color reversible): La transformación se realiza en el dominio de números enteros, en el que la operación puede ser invertida sin pérdida de precisión.

Transformación espacial

Consiste en la división de la imagen en tiles[1]. Éstos pueden ser de cualquier tamaño aunque todos del mismo, excepto quizá los tiles que se sitúen en los bordes de la imagen. La DWT, llamada transformación espacial en la Fig. 1.4, se aplica a los tile-components[2] resultantes de la etapa anterior. Éstos se transforman y codifican por separado, por lo que: (1) el descodificador necesitará menos memoria al descodificar sólo las tiles seleccionadas, y (2) se pueden aplicar parámetros distintos a cada uno de ellos.

Cuantificación

Después de la transformada wavelet, los coeficientes se escalan para reducir el número de bits necesarios para representarlos, a expensas de la calidad. La salida es un conjunto de números enteros que deben ser codificados en planos de bits. El parámetro que se puede modificar para establecer la calidad final es el paso de cuantificación (Δ_s) que se aplica mediante:

$$q_s[n] = sign(y)\left\lfloor \frac{|y_s[n]|}{\Delta_s} \right\rfloor. \tag{1.4}$$

Como puede apreciarse en esta expresión, cuanto mayor es el paso, mayor es la compresión y la pérdida de calidad. Con un paso de cuantificación igual a 1, no se realiza ninguna cuantificación (se utiliza en la compresión sin pérdida). Por tanto, es un proceso que reduce la precisión de las muestras $y_s[n]$a un conjunto discreto de niveles generando coeficientes cuantificados $q_s[n]$ en la subbanda ss. El parámetro sign(y) denota el signo de y, e $|y|$ su valor absoluto.

[1]El estándar JPEG2000 permite dividir una imagen en pequeñas regiones rectangulares llamadas tiles (en español podría traducirse como baldosa), los cuáles se comprimen independientemente al resto, por lo que los parámetros de compresión pueden ser distintos para cada uno.

[2]En una imagen de tres componentes un tile-component está formando por tres tile-components, uno por componente, a los que a cada uno se le aplica el proceso de compresión de forma independiente.

Este proceso puede llevarse a cabo con o sin pérdida:

1. *DZQ*: la etapa DeadZone Scalar Quantization, en español cuantificación escalar de zona-muerta, que es irreversible porque introduce ruido de cuantificación al producir índices cuantificados aproximados, debido a que trabaja sobre valores no enteros.

   Los índices caen en un rango ("dead zone") donde no se distinguen unos de otros por falta de precisión. La zona muerta sirve para que aumente la cantidad de índices iguales a cero ($q_s[n] = 0$) tras cuantificar. La disminución de información aumenta la compresión a un coste en la distorsión relativamente bajo, dado que el sistema visual humano es poco sensible a la pérdida de información visual de baja intensidad.

2. *Ranging*: Se utiliza en la codificación sin pérdida donde $\Delta_s = 1$. Las muestras son cuantificadas por valores enteros, por lo que la salida no requiere redondeo y por tanto, no se genera ruido o errores de cuantificación.

ROI

La siguiente etapa es la definición de las llamadas ROI (Region of Interest, en español región de interés) es una región definida en tiempo de compresión y puede tener cualquier forma. En caso de existir, los code-blocks[1] incluidos en dicha región se extraerán en mayor medida que el resto, proporcionando una mayor calidad de los mismos.

VLC

Codificación de Longitud Variable (Variable Length Coding, en inglés), hace posible la compresión sin pérdida mediante la codificación por planos de bits, con truncado óptimo usando EBCOT (Embedded Block Coding with Optimal Truncation, en español codificación de bloques embebidos con truncamiento óptimo). EBCOT se basa en la codificación independiente de code-blocks relativamente pequeños de

[1] Cada tile-component es dividido en code-blocks, que son codificados independientemente.

muestras de las subbandas. Por lo tanto, comprime los coeficientes wavelet, generando un bit-stream (con escalabilidad en calidad) embebido para cada code-block de forma independiente. Para esto se utiliza un codificador por planos de bits que realiza tres pasadas sobre cada plano de bits de cada code-block, resultando un bit-stream que puede ser truncado. Estos *coding-passes*, en español pasos de codificación, incrementan la escalabilidad en calidad, añadiendo más puntos de truncado al code-stream. La división en code-blocks es fundamental para el paradigma EBCOT, empleado en el estándar JPEG2000.

PCRD-opt

Por último, la etapa de optimización R/D en tiempo de post-compresión organiza los datos del code-stream para lograr las diferentes formas de escalabilidad y minimizar la distorsión dada una tasa de bits objetivo. PCRD utiliza para ello cada punto de truncado del code-stream proporcionado por: capas de calidad, resolución, components, precints[1] y los planos de bits dados por los coding-passes en los code-blocks.

Se definen 5 tipos de organización de los datos que componen las imágenes. La ordenación que se lleve a cabo no afecta a los paquetes en sí, éstos no se modifican.

1. LRCP (Layer-Resolution-Component-Precint): Esta progresión es principalmente progresiva por calidad. Durante la descompresión para la escalabilidad en calidad, las subbandas deben ser descodificadas en orden por capas de calidad, resolución, componentes y precintos. En este orden los paquetes tienen una mejor relación entre el tamaño mínimo del code-stream truncado y la distorsión de la imagen descomprimida.

   Cada capa de calidad consiste en un conjunto de code-blocks (cada uno de ellos con varios puntos de truncado gracias a los coding-passes). Los code-blocks son seleccionados para provocar una disminución significativa de la distorsión en la reconstrucción, ya que puede haber paquetes vacíos o con un impacto insignificante en la distorsión. Por lo tanto, la elección del número de planos de bits a incluir en una capa de calidad para la reconstrucción, se basa en su tasa de rate/distorsión (R/D).

[1] Los code-blocks son agrupados en precints (en español, precintos).

2. RLCP: Progresión principalmente progresiva por resolución. La progresión dentro de una resolución dada, es por capa de calidad.

3. RPCL: Progresión progresiva por resolución y dentro de una resolución, progresiva por "precint".

4. PCRL: Progresión progresiva espacialmente.

5. CPRL: Progresión por "component".

1.2.2 Características de JPEG2000

Como se ha expuesto JPEG2000 al igual que otros compresores de imagen, se basa en la cuantificación escalar de los píxeles de la imagen en el dominio wavelet. Sin embargo, JPEG2000 es especialmente interesante para la compresión de vídeo debido a su:

1. Rendimiento de codificación superior con respecto a JPEG, especialmente en bit-rates muy bajos.

2. Posibilidad de codificación con o sin pérdida. La codificación con pérdida es más eficiente en términos de rate/distorsión, ya que la relación entre la cantidad de datos y la distorsión en la reconstrucción no es lineal (especialmente en el caso de la distorsión percibida por el ojo humano). Es decir, se puede reducir el bit-rate del code-stream en gran medida, produciendo una distorsión subjetiva relativamente baja. La ruta sin pérdidas, al tener una distorsión nula en la reconstrucción es usada en ámbitos como la medicina.

3. Generación de code-streams escalables en calidad y espacialmente. Además el número de capas de calidad puede ser muy alto y la sobrecarga introducida es relativamente pequeña.

4. Capacidad de descomprimir sólo una región de interés de la imagen, lo cual es muy útil en aplicaciones de navegación remota, ya que sólo es necesario transmitir los paquetes J2K (J2K-packets)[1] relacionados con la WOI.

[1] Este es el nombre que el estándar da al tamaño mínimo de paquete que puede ser descomprimido.

Los code-stream JPEG2000 son escalables (1) en resolución, dependiendo del número de niveles de la 2D-DWT, (2) en calidad, dependiendo del número de capas de calidad disponibles, y (3) en espacio (considerando la posibilidad de usar WOIs).

1.3 Codificación de vídeo escalable

La comunidad ha investigado la codificación de vídeo escalable [28, 42] (SVC, en inglés Scalable Video Coding) durante décadas. La tasa de bits necesaria para transmitir vídeo sin comprimir es demasiado alta. Por ese motivo, la gran mayoría de los vídeos están codificados de acuerdo a una representación que requiere menos bits para su almacenamiento y transmisión. Además, los servidores para streaming de vídeo necesitan transmitir datos a una amplia variedad de terminales, posiblemente con diferentes resoluciones de pantalla, diferentes capacidades de computación y a través de redes con un ancho de banda variable. La codificación de vídeo escalable es una alternativa para resolver de forma óptima el streaming de vídeo bajo estas restricciones.

Por otro lado, el rate-control en relación a la calidad de imagen, se ha convertido en uno de los retos más importantes en la transmisión de vídeo, ya que su tratamiento adecuado maximiza la calidad de los vídeos reconstruidos. Ésta ha sido la motivación para que varios códecs de vídeo estándar hayan sido propuestos por la ISO/IEC (International Electrotechnical Commission) que propuso MPEG-* (Moving Picture Experts Group), y la ITU que desarrollo H.26*. Algunos de estos estándares, que en este caso han sido propuestos conjuntamente por ambas organizaciones, es la extensión escalable del estándar H.264/SVC [51] y H.265/AVC (Advanced Video Coding, en español codificación de vídeo avanzada), denominado SHVC (Scalable High-efficiency Video Coding) [57].

Para generar un code-stream escalable, necesitamos usar codificadores de vídeo escalables. Un SVC produce un único code-stream escalable, organizado de tal manera que varias versiones del mismo vídeo puedan ser reconstruidas en la etapa de descodificación utilizando sólo una parte del code-stream, y de forma acumulativa, a mayor descodificación del code-stream, mayor calidad, resolución o imágenes/segundo tendrá la reconstrucción.

Mediante el uso de un SVC, podemos desacoplar los procesos de compresión y descompresión, lo que será interesante para los sistemas de streaming de vídeo, donde

los usuarios pueden ver el vídeo descodificado tan pronto como se reciba. Por ejemplo, el lado servidor, puede alojar un vídeo de alta resolución, con una alta frecuencia de imágenes/segundo y nivel de calidad. Si un usuario solicita dicho vídeo pero con menores especificaciones, no es necesario realizar una nueva codificación, basta con entregar un subconjunto del code-stream escalable que satisfaga los requisitos del usuario.

1.3.1 Codificación de imágenes en un vídeo

La codificación de vídeo que explota la correlación temporal entre imágenes es aplicada a nivel de grupos de imágenes, en inglés Group of Pictures (GOP), cada uno de ellos contiene un determinado número de imágenes (tamaño de GOP G). Cada imagen (V_i) de un GOP se codifica según la correlación temporal encontrada, así existen distintos tipos de imágenes y agrupaciones de las mismas, descritas a continuación.

1.3.1.1 Tipos de imágenes en un vídeo

Dependiendo de cómo se construyan las predicciones se encuentran los siguientes tipos de codificación de imágenes y simetría de los GOPs:

1. *Intra-codificada:* Una imagen intra (I) es aquella en la que su codificación depende únicamente de sí misma y no de otra/s. Es decir, no es necesario realizar una estimación de movimiento y por lo tanto, no contiene información sobre la correlación temporal.

2. *Predicha-codificada:* Las imágenes llamadas P son aquellas cuya codificación depende de la imagen anterior (en la secuencia de imágenes). La imagen de la que depende P puede ser otra imagen P o una imagen I.

3. *Bidireccionalmente predicha-codificada:* La dependencia de una imagen B puede ser respecto de imágenes I o B. Una imagen B depende de dos imágenes (una imagen P depende sólo de una). Por lo tanto, la dependencia es bidireccional es decir, depende de una imagen anterior y una posterior en la secuencia de imágenes. Estas dependencias se representan en forma de vectores de movimiento (MVs).

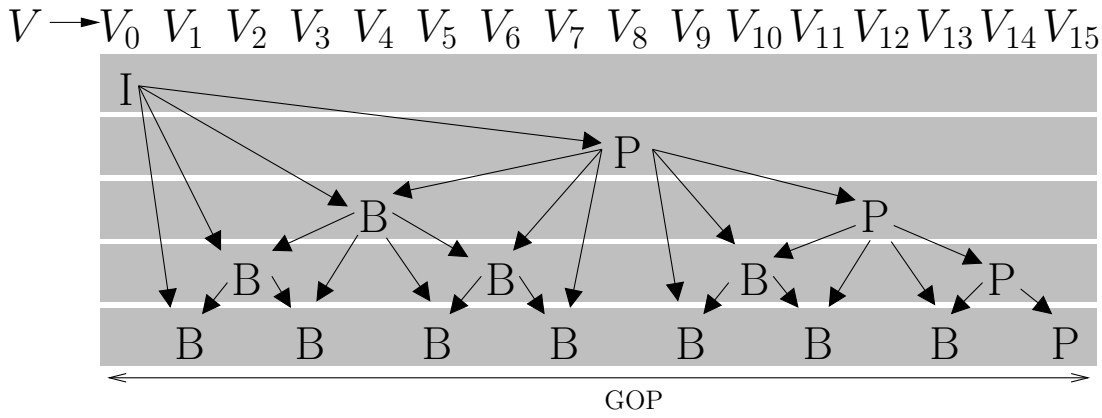


Figura 1.5: Dependencias de escalabilidad temporal en un GOP cerrado y asimétrico.

1.3.1.2 **Tipos de GOPs**

Los GOPs, según su dependencia respecto de otros durante su reconstrucción, se pueden clasificar en:

1. *GOPs cerrados*: se dice que un GOP es cerrado cuando se puede descodificar (y visualizar) independientemente de cualquier otro GOP. En el orden de visualización, los GOPs cerrados comienzan necesariamente con una imagen I. Ejemplo: *I B B P B B P B B P B B P*.

2. *GOPs abiertos*: Un GOP abierto necesita al menos de un GOP vecino para descodificarse. Los GOPs abiertos suelen empezar (respecto del orden de visualización) en una imagen B. Estos GOPs tienen un rendimiento R/D menor en el acceso aleatorio a imagen que los códecs que usan GOPs cerrados. Esto se debe a que para un mismo tamaño de GOP, un GOP cerrado se descodificará en base a un menor número de imágenes, necesitando un rate menor. Por otro lado, los GOPs abiertos no limitan los errores de predicción en el tiempo a nivel de GOP, propagándose dichos errores de deriva hasta que la descodificación dependa de otras imágenes. Ejemplo: *B B I B B P B B P B B P B B P*.

En las siguientes figuras se muestran las dependencias para la escalabilidad temporal de GOPs con distinta simetría. En la Fig. 1.5 se muestra un ejemplo de GOP cerrado y asimétrico. En la Fig. 1.6 se expone una representación de las dependencias de la escalabilidad espacial de un GOP cerrado y asimétrico, cuando se usan dos niveles de

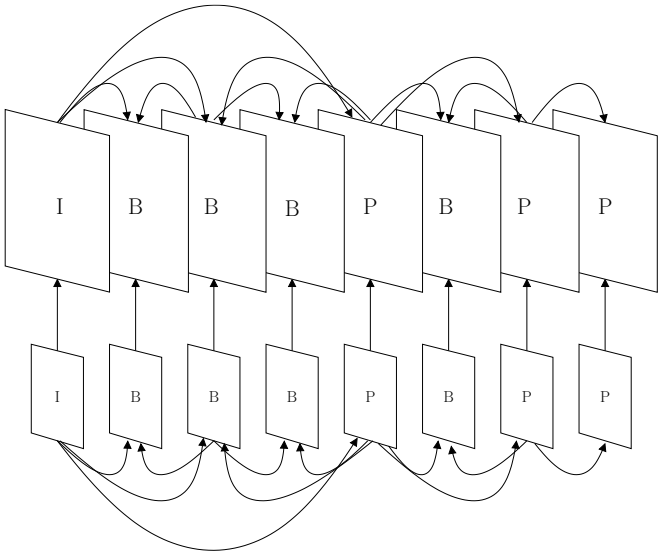


Figura 1.6: Dependencias de escalabilidad espacial en un GOP cerrado y asimétrico.

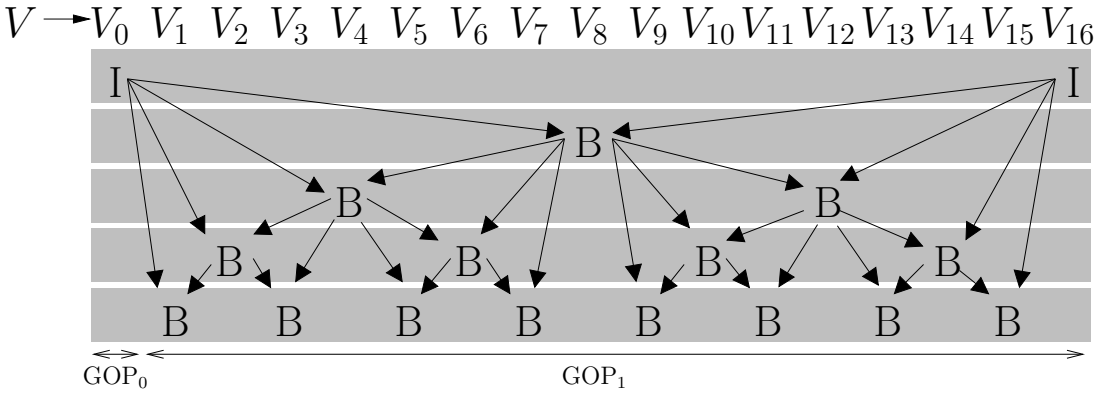


Figura 1.7: Dependencias de escalabilidad temporal en un GOP abierto y simétrico.

resolución espacial (generados mediante una pirámide Laplaciana). En la Fig. 1.7 se muestra un ejemplo de GOP abierto y simétrico.

1.3.2 Motion JPEG2000

El comité JPEG amplió el estándar JPEG2000 [22] para la codificación de vídeo. El resultado se conoce como MJ2K (Motion JPEG2000) [20], que tiene como objetivo generar un vídeo comprimido altamente escalable que pueda ser fácilmente editado. MJ2K es una extensión directa de J2K donde cada imagen de una secuencia de vídeo se comprime independientemente (imágenes intra), y por lo tanto, los code-streams MJ2K son totalmente escalables en el dominio temporal dado que cada imagen puede ser descodificada sin descodificar ninguna otra. En contrapartida, MJ2K no incluye compensación de movimiento, por lo que no explota la correlación temporal entre imágenes. Por esto sus ratios de compresión son bastante modestos en comparación a códecs como AVC o SHVC, que son mucho más eficientes desde el punto de vista de rate/distorsión, gracias a la descorrelación temporal que se utiliza para eliminar la redundancia temporal de la secuencia de imágenes. Sin embargo MJ2K aún se utiliza en algunos entornos como en la industria de distribución de cine digital [7].

1.3.3 Codificación híbrida de vídeo (DPCM)

La mayoría de los estándares son codificadores con pérdida de información (las reconstrucciones no son idénticas a las originales) y se basan en la idea de la codificación de vídeo predictiva con pérdida de bucle cerrado (DPCM) [68], consulte la Fig. 1.8. El códec produce una secuencia de imágenes reconstruidas $V_i^{[q]}$ que son aproximaciones a las originales V_i dependiendo del paso de cuantificación q, donde i indica el número de imagen en la secuencia. El codificador transmite al descodificador una secuencia de residuos de imágenes cuantificadas $E_i^{[q]}$ que tienen una cantidad de datos menor que las imágenes originales de V y por lo tanto, puede ser comprimido en mayor medida.

El algoritmo de estimación y compensación de movimiento utiliza la imagen reconstruida anterior $V_{i-1}^{[q]}$ como referencia para la actual V_i. Observe entonces que para controlar el error de deriva con respecto al descodificador, el codificador utiliza las

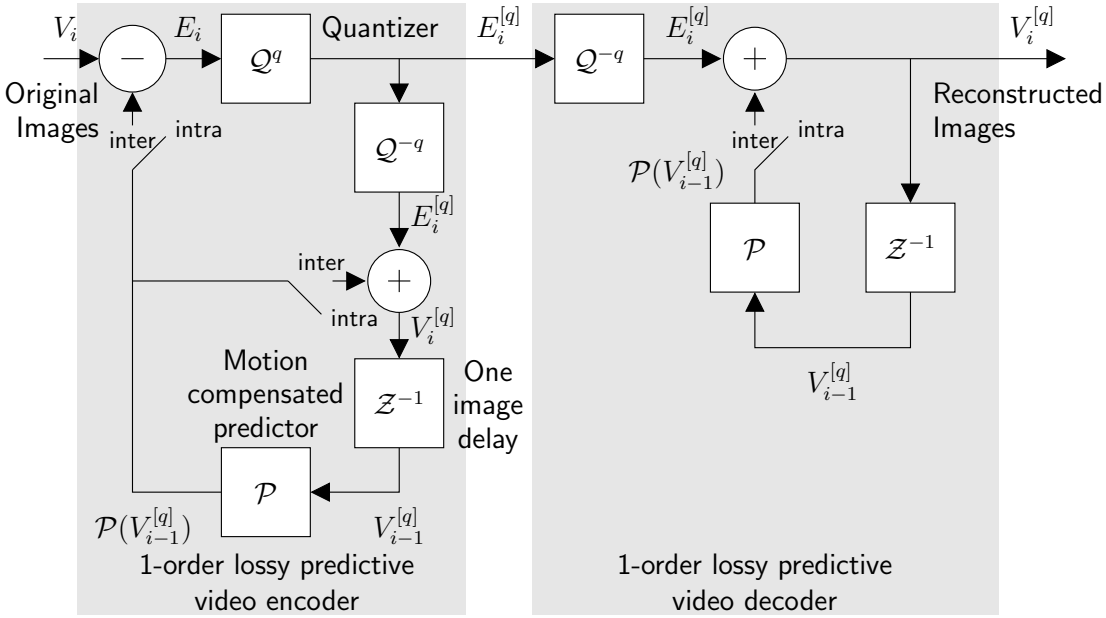


Figura 1.8: Esquema básico de un códec de vídeo con pérdida de bucle cerrado (DPCM) basado en la compensación de movimiento.

imágenes $V_i^{[q]}$ (y no las imágenes originales), para construir las predicciones. Es decir, el bucle cerrado supone realimentar las predicciones con los errores de imágenes anteriores para reducir el error en la imagen actual.

De esta forma las imágenes del vídeo se codifican secuencialmente utilizando un esquema IPPP (véase arriba en la Fig. 1.15) a nivel de GOP, donde la letra I denota una imagen intra-codificada y la letra P representa una imagen codificada como una predicción que depende de la imagen descodificada anteriormente.

1.3.3.1 **DPCM en códecs de vídeo**

Los estándares MPEG e ITU se basan en un esquema de bucle cerrado porque son más eficientes desde el punto de vista del nivel de compresión alcanzado que los esquemas de lazo abierto. Además, para las extensiones escalables (H.264/SVC y SHVC) también se siguió dicho diseño. En dichas extensiones la escalabilidad espacial utiliza una pirámide Laplaciana por cada imagen y la escalabilidad en calidad es un caso especial de escalabilidad espacial donde las resoluciones permanecen constantes. Como conse-

cuencia directa, el número de capas de calidad se reduce drásticamente, pero el grupo MPEG/ITU para compensarlo propone un *extractor de bits* que aumenta considerablemente la granularidad del code-stream, hasta el punto de que el último inconveniente puede ser ignorado en la mayoría de escenarios.

1.3.3.2 **H.264/AVC**

H.264/AVC [51, 65] es un códec de vídeo estándar que proporciona una alta eficiencia de compresión, compatibilidad de red y una amplia gama de posibilidades en cuanto a resistencia a errores. La versión escalable de H.264/AVC es H.264/SVC.

El codificador de vídeo H.264 realiza procesos de predicción, transformación y codificación. El descodificador realiza el proceso invertido. Ambos se describen a continuación.

1. *Codificador*: El proceso de codificación comienza con la estimación de movimiento. Delimita macro-bloques de tamaño variable, desde 4×4 hasta 16×16, esto permite una compensación de movimiento más precisa, ya que se adapta a las características del vídeo codificado. El movimiento relativo de los macro-bloques de la imagen se codifica con respecto a una o varias imágenes anteriores ya codificadas.

2. *Transformación y cuantificación*: Se aplica la transformada DCT (Discrete Cosine Transform) en los macro-bloques de los residuos (en una transformación de 4×4 o 8×8). Esta transformación genera un conjunto de coeficientes, cada uno de los cuales es un valor de ponderación para un patrón base estándar. Luego se cuantifica los coeficientes, es decir, cada coeficiente se divide por un valor entero. Esto da como resultado una pérdida de datos según la cuantificación. Una mayor cuantificación significa que más coeficientes resultan en cero, es decir, una alta compresión a costa de una mayor pérdida de información, lo que resultará en una reconstrucción con mayor distorsión. Un valor bajo tiene el efecto opuesto, más coeficientes distintos de cero que producirán una reconstrucción con menos distorsión.

3. *Codificación del code-stream*: El proceso de codificación produce un conjunto de datos que una vez codificados forman el code-stream, el cual principalmente contiene:

   (a) Coeficientes de cuantificación.

   (b) Información para que el descodificador reconstruya la predicción de movimiento.

   (c) Información sobre la estructura del code-stream.

   Todos estos datos se empaquetan utilizando la codificación de longitud variable y/o la codificación aritmética. Éstos producen una representación binaria eficiente de la información.

4. *Descodificación del code-stream*: Cuando el descodificador recibe el code-stream, descodifica la información descrita anteriormente invirtiendo el proceso de codificación y reconstruye la secuencia de imágenes. Los coeficientes cuantificados se reescalan multiplicando por el valor entero que se usó en la cuantificación y así restaurar su escala original. El valor resultante es diferente al original, ahora incalculable. Más tarde, una transformación inversa combina los patrones básicos estándar, para reconstruir cada macro-bloque de datos residuales.

5. *Reconstrucción*: Para cada macro-bloque, la predicción de movimiento se recrea sin pérdida de información (los MVs reconstruidos deben ser los mismos que los codificados, ya que su falta de precisión provoca una reconstrucción errática). Posteriormente la predicción y el residuo descodificado se combinan, formando un macro-bloque descodificado.

1.3.3.3 **HEVC**

HEVC (High-Efficiency Video Coding, en español “codificación de vídeo de alta eficiencia”) es la evolución del códec H.264/AVC. Las principales mejoras de éste son:

1. *Codificación de unidades de árbol*: HEVC reemplaza los macro-bloques por unidades de árbol, en inglés Coding Tree Units (CTU). La ventaja es que estas unidades pueden ser más grandes, hasta 64×64 píxeles, además hace uso de un

tamaño de bloque variable [10] para ajustarlo dinámicamente a las características del movimiento en cada codificación. Este mayor rango de tamaños para dividir la imagen en unidades que pueden ser localizadas entre las imágenes de referencia posibilita una estimación de movimiento más precisa.

2. *Predicción*: El cálculo de los MVs es ahora más preciso, se puede codificar el desplazamiento de una unidad del árbol en un rango más amplio de direcciones (33 orientaciones diferentes). Mayor precisión al determinar el movimiento en el vídeo hace que la estimación de movimiento sea mejor, lo que produce un residuo más pequeño.

1.3.4 Filtrado temporal con compensación de movimiento (MCTF)

MCTF (en inglés, Motion Compensated Temporal Filtering) es una técnica para codificación de subbandas temporales que se aplica a secuencias de imágenes (GOP a GOP). MCTF devuelve a una secuencia de subbandas temporales e información adicional sobre la compensación de movimiento en forma de vectores de movimiento (MVs).

En la Fig. 1.9, el vídeo original se divide en subbandas temporales. A partir de la última imagen del GOP actual, la última del GOP anterior (ambas imágenes intra) y la estimación del movimiento, se calculan las imágenes residuales (B). Una imagen B se compone entonces de residuos, es decir, de las texturas que no pudieron ser predichas por la estimación de movimiento a partir de las imágenes de referencia.

MCTF no usa un bucle en el codificador para cancelar la desviación entre su predicción y las producidas en el descodificador y por tanto este error de deriva existe. Sin embargo, debido a la forma en que las tramas se procesan, este error no se acumula a lo largo del tiempo y además está acotado a nivel de GOP. Esto se debe a que todas las predicciones en un GOP están basadas en última instancia de imágenes intra.

1.3.4.1 Implementación mediante un banco de filtros

MCTF puede implementarse eficientemente como una cascada diádica de filtros paso-bajo (L) y paso-alto (H) que generan un conjunto de subbandas temporales $\{L^{T-1}, H^{T-1}, \cdots, H^1\}$, siendo T el número de niveles de resolución temporal. Una

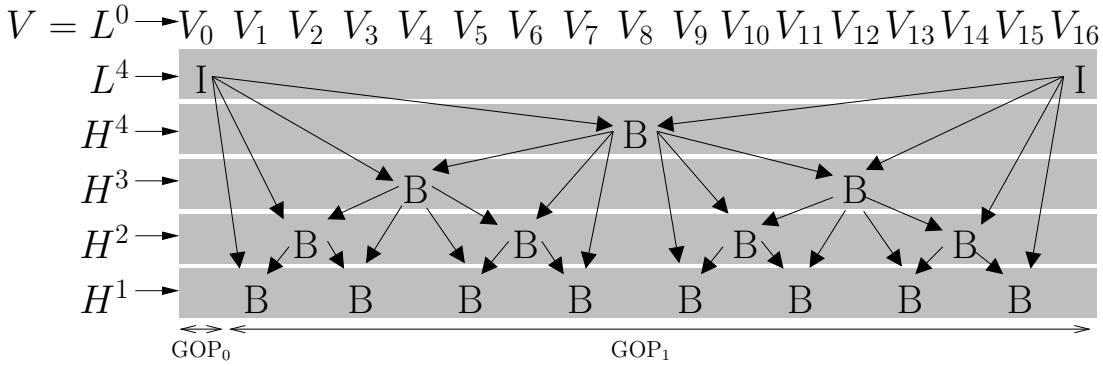


Figura 1.9: Dependencias de escalabilidad temporal en un GOP abierto y simétrico. Ejemplo de MCTF$^{T=4}$ ($G = 2^T$).

muestra en una subbanda es una imagen o un residuo, dependiendo de la subbanda (véase la Fig. 1.10).

Esta transformación es totalmente reversible porque sostiene que

$$\begin{aligned} S &= \uparrow^2 (L^1) * \gamma_L + \uparrow^2 (H^1) * \gamma_H, \\ L^1 &= \uparrow^2 (L^2) * \gamma_L + \uparrow^2 (H^2) * \gamma_H, \\ &\vdots \\ L^{T-1} &= \uparrow^2 (L^T) * \gamma_L + \uparrow^2 (H^T) * \gamma_H, \end{aligned} \tag{1.5}$$

donde $\uparrow^2$ denota el operador de sobre-muestreo (en un factor de 2), $*$ es la convolución de dos señales discretas, y $\{\gamma_L, \gamma_H\}$ es el banco de filtros de síntesis utilizado en la transformación MCTF inversa de nivel 1 (denotado por MCTF^{-1}).

1.3.4.2 Lifting en MCTF

La Fig. 1.11 muestra cómo se ha implementado una de las etapas de la cascada usando Lifting [60], mostrando en detalle un paso de la transformación de un nivel temporal, directo e inverso.

Las tres etapas de codificación son las siguientes:

1. La etapa "Split" separa la señal en dos conjuntos disjuntos de imágenes, las pares y las impares. Este proceso de separación también es llamado Lazy wavelet transform en inglés. Siendo en la figura L_i^t la imagen número i (dentro de la secuencia de imágenes) de la subbanda L, del nivel de resolución temporal t.

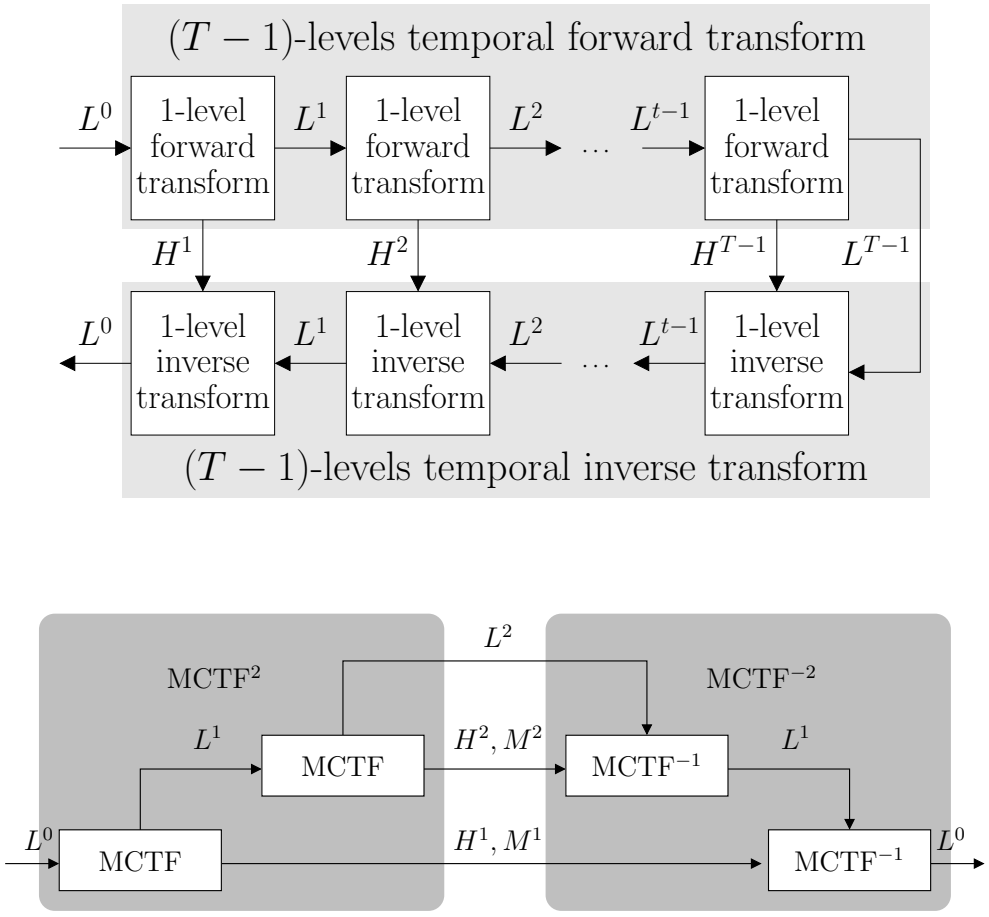


Figura 1.10: Transformación temporal usada en MCTF para T niveles (arriba) y un ejemplo donde $T = 2$ (abajo).

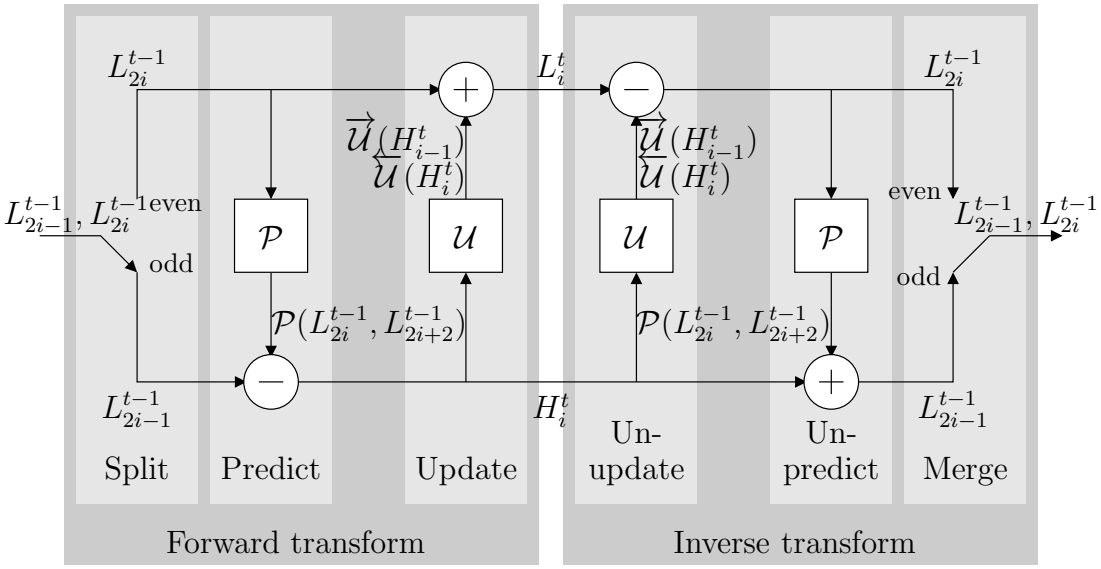


Figura 1.11: Implementación de un paso de la transformación temporal utilizando Lifting.

2. La etapa de predicción se basa en que los subconjuntos pares e impares están intercalados. Si la señal tiene correlación local, los subconjuntos pares e impares estarán altamente correlacionados. En otras palabras, dado uno de los dos conjuntos, debería ser posible predecir el otro con una precisión razonable. Siempre se usa el conjunto par para predecir el impar, es decir, la predicción se realiza respecto de las imágenes pares.

3. Estas predicciones casi nunca son perfectas. Por esto, se lleva a cabo la etapa de actualización para minimizar el efecto de aliasing producido por el sub-muestreo temporal de la secuencia de vídeo sin aplicar previamente un filtro paso-bajo. Las imágenes residuo resultantes constituyen la resta de la imagen original menos la predicha, quedando almacenado como imagen de la subbanda H esta diferencia.

Todo el proceso puede simplificarse en este ejemplo, sean dos señales A y B, media E y diferencia D. Se puede recuperar A y B a partir de E y D. Cuanto mayor sea la correlación entre las dos señales, menor será el valor de D y menos bits serán necesarios para su representación. Observe además que la transformación inversa usa el

residuo y la predicción para recuperar la imagen original. Si la compresión del residuo se hizo con pérdida, la imagen recuperada es una aproximación de la imagen original.

1.3.4.3 DWT y MCTF

La mayoría de las técnicas SVC propuestas se basan en la transformada wavelet discreta 3D con compensación de movimiento [9, 13, 29, 45, 46, 54]. En todas estas propuestas, la descorrelación temporal (DWT 1D) se realiza antes que la espacial (DWT 2D), por ello estas técnicas también se conocen como códecs de vídeo t+2D. La principal ventaja de un algoritmo t+2D es que el uso de cualquier procedimiento estándar de compensación de movimiento es sencillo porque se aplica al dominio de la imagen (no transformado). Esencialmente, las técnicas de t+2D son equivalentes a MCTF+2D.

Por otro lado, también se pueden utilizar las técnicas 2D+t [2, 8, 16]. Éstas realizan la MC (motion compensation, en español compensación de movimiento) en el dominio wavelet, aunque este dominio no es invariante al desplazamiento y, por lo tanto, la estimación de movimiento tiene que aplicarse en el dominio redundante de la DWT. La principal ventaja de un algoritmo 2D+t sobre uno t+2D es que, si la MC se aplica para cada resolución espacial, entonces la información del movimiento tendrá una representación multiresolución, y ésta puede ser aplicada a cada nivel de resolución espacial de las imágenes para minimizar los requerimientos de memoria y cálculo en el descodificador, según la versión a resolución reducida que el cliente recibe. Sin embargo, hacer una MC para cada resolución espacial tiene otras dos desventajas:

1. Si durante la transmisión, los requerimientos o recursos del cliente como el ancho de banda cambian y es necesario aumentar o disminuir la cantidad de datos a transmitir, será necesario reenviar los MVs de nuevo para el GOP actual, o esperar a que el GOP actual termine, y enviar los MVs de la resolución correcta a partir de ahí.

2. Se sobredimensiona el espacio ocupado por los MVs en el code-stream, ya que habrá un conjunto de ellos para cada resolución espacial.

1.3.4.4 Code-stream resultante de MCTF

El resultado de aplicar MCTF a una secuencia de imágenes es otra secuencia de imágenes filtrada en el dominio del tiempo, que puede ser comprimida eficientemente por un

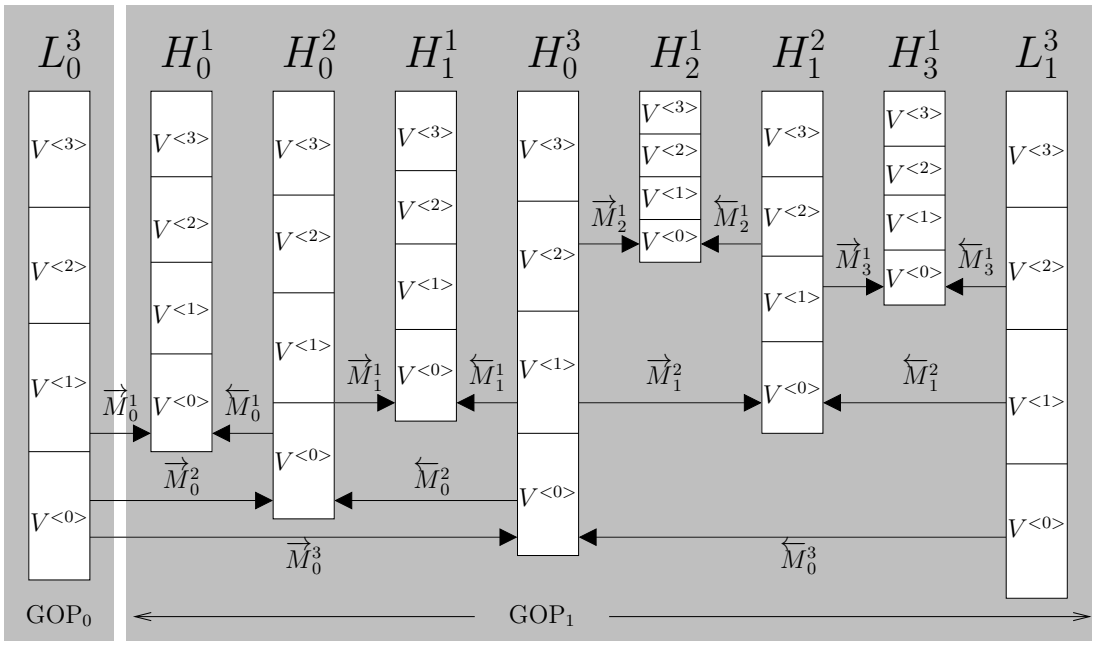


Figura 1.12: Compresión de un vídeo de nueve imágenes que proporciona escalabilidad temporal y espacial.

compresor de imágenes como MJ2K. El proceso MCTF devuelve un conjunto de subbandas temporales, véase la Fig. 1.12, donde se muestra la codificación por MCTF de un vídeo de nueve imágenes. Se generan tres subbandas H y una L, en ellas se distribuyen cada una de las nueve imágenes $\{L_0^3, H_0^1, H_0^2, H_1^1, H_0^3, H_2^1, H_1^2, H_3^1, L_1^3\}$ donde s_i^t indica el número de imagen (i) dentro de la subbanda (s) de nivel (t) $\forall\, t \in \{T, \cdots, 1\}$, siendo T el número de niveles de resolución temporal ($TRLs$).

Todas las imágenes en la Fig. 1.12 contenidas en una subbanda H son imágenes predichas ya que dependen de MVs para su reconstrucción, tal y como indican las flechas de los campos de movimiento (M en la figura). Por ejemplo, la imagen H_0^1 es predicha a partir de las imágenes adyacentes L_0^3 y H_0^2. Nótese que la imagen H_0^2 a su vez está predicha a partir de otras.

Con objeto de explicar totalmente la Fig. 1.12, después de la etapa MCTF, se comprimen todas estas subbandas y los MVs resultantes asociados a cada imagen. Se usará un compresor de imagen escalable que proporcionará un code-stream de cuatro capas para la escalabilidad espacial. Éstas se ordenan desde la resolución más baja

Figura 1.13: Escalabilidad temporal en un vídeo.

(representada en la figura por la capa de calidad R_3) hasta la más alta (R_0). La información de movimiento se codifica usualmente sin pérdidas, evitando imprecisiones en los vectores de movimiento.

1.3.4.5 **Escalabilidad temporal restringida por MCTF**

MCTF limita la escalabilidad temporal. En los códecs que no explotan la correlación temporal, como MJ2K, la escalabilidad temporal tiene la máxima granularidad, de modo que cualquier imagen puede ser descomprimida sin necesidad de datos de otras imágenes. En los códecs que sí explotan la correlación temporal, la escalabilidad temporal tiene menos granularidad por este mismo motivo, y en consecuencia, tienen restricciones sobre qué conjuntos de imágenes pueden ser descomprimidos. Usando MCTF estos conjuntos son temporalmente equidistantes (véase la Fig. 1.13). En este esquema definimos un nivel de resolución temporal t como

$$V^t = \{V_{2^t \times i};\ 0 \leq i < \frac{\#V}{2^t}\} = \{V_{2i}^{t-1};\ 0 \leq i < \frac{\#V^{t-1}}{2}\}, \tag{1.6}$$

donde $\#V$ es el número de imágenes en V. Nótese que $V = V^0$.

Figura 1.14: Cruce de datos de múltiples imágenes (izquierda) y sin cruce (derecha).

1.3.4.6 **MCTF en códecs de vídeo**

Numerosos estudios aplican alguna estrategia de predicción de movimiento utilizando técnicas como MCTF [17] o LIMAT [54] (Lifting-based Invertible Motion Adaptive Transform, en español transformación adaptativa de movimiento invertible basada en elevación), y que muestran resultados interesantes.

Existen implementaciones de MCTF que utilizan múltiples imágenes [63], por lo que el número de imágenes de referencia a partir de las cuales encontrar macro-bloques similares es mayor y por tanto, la cantidad de MC que se pueda realizar también lo será. Sin embargo, algunos trabajos [40, 67] indican los problemas que conlleva en términos de rendimiento y calidad visual de las imágenes. Uno de los problemas más llamativos es el denominado efecto fantasma (Fig. 1.14) que se produce por el hecho de relacionar dos o más imágenes entre sí y que da lugar a imágenes en los que se muestra información de varios momentos simultáneamente. Existen trabajos en los que estas interferencias se pueden reducir mediante el uso de técnicas aplicadas a subbandas temporales de paso bajo [36] y que mejoran sustancialmente la escalabilidad temporal y espacial.

Otro enfoque es el uso de imágenes intra para controlar de manera efectiva la deriva de la calidad de la codificación, basada en paquetes escalables con estructuras de predicción jerárquicas [52]. Este proceso también puede aplicarse a partes de la imagen que no están en movimiento, utilizando métodos de adaptación de contenido para la compensación de movimiento tridimensional [35].

	HEVC	*SHVC*	*SVC*	*MCJ2K*	*MJ2K*
Eficiencia de compresión	100 %	80 %	50 %	50 %	10 %
Escalabilidad temporal	Sí	Sí	Sí	Sí	Sí
Acceso aleatorio a imágenes	No	No	No	No	Sí
Escalabilidad espacial	No	Sí	Sí	Sí	Sí
Escalabilidad en calidad	No	Sí	Sí	Sí	Sí
Codificación ROI	No	No	No	Sí	Sí
Escalabilidad WOI	No	Sí	No	Sí	Sí
Descodificación interactiva	No	No	No	Sí	Sí
Tiempo de codificación	Lento	Muy lento	Rápido	Rápido	Muy rápido
Tiempo de descodificación	Rápido	Rápido	Rápido	Rápido	Rápido
Máx. imágenes por segundo	300	300	60	-	-
Máx. resolución	8K	8K	4K	64K	64K
Máx. profundidad de bits	16	16	14	32	32

Tabla 1.1: Comparativa de características de códecs.

Existen otros trabajos que a través de una compresión de vídeo altamente escalable [54] optimizan cuantitativamente el número de cálculos basados en la transformada wavelet, incrementando significativamente la granularidad de la escalabilidad del code-stream, con un detrimento relativamente pequeño en la calidad.

En la Tab. 1.1 hay una comparativa de las características de los códecs que se utilizan comúnmente hoy en día. Todos ellos utilizan algún tipo de técnica que reduce la correlación temporal, excepto MJ2K.

1.3.5 DPCM vs. MCTF

La Fig. 1.15 muestran las dependencias entre imágenes producidas por el algoritmo de bucle cerrado (arriba en la figura), que produce el esquema IPPP. Y su correspondencia con las dependencias de las imágenes en MCTF (abajo en la figura). Observe que en MCTF una distorsión (o error) introducida en una imagen no se propaga linealmente a lo largo del tiempo. Por esta razón, el bucle en el codificador puede ser eliminado.

La distribución de las imágenes en los GOPs difiere entre ambos algoritmos. La técnica de bucle cerrado en la Fig. 1.15 presenta que el GOP_0 contiene las imágenes $\{I_0, P_1, \cdots, P_7\}$, el GOP_1 se compone de la imagen I_8. La primera imagen de cada GOP está codificada sólo en base a sus texturas con J2K, por lo que no depende de ninguna otra. El resto son imágenes predichas.

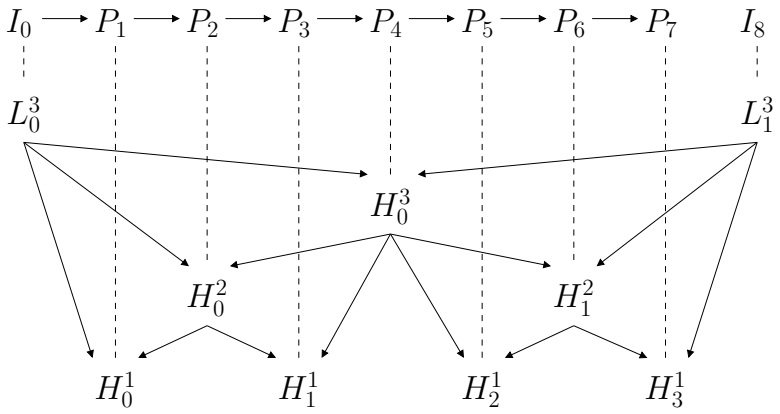


Figura 1.15: Dependencias entre imágenes en una técnica de bucle cerrado (arriba) y en MCTF (abajo).

En la técnica MCTF, la distribución de imágenes en los GOPs se lleva a cabo entre subbandas L y H (contienen bajas y altas frecuencias, respectivamente). En la Fig. 1.15 la distribución es la siguiente: el GOP_0 sólo contiene una imagen L_0^3, el GOP_1 incluye el resto de las imágenes $\{H_0^1, \cdots, L_1^3\}$. En MCTF, todas las imágenes de subbandas L son intra (I); en las subbandas H, se pueden encontrar imágenes I (no-predichas) como B (predichas).

1.3.6 **Rate allocation**

El término rate-allocation (RA) se refiere al proceso que decide qué información codificar cuando existe un límite en el "rate", con el objeto de obtener la mejor relación R/D posible. Se formula como un problema de optimización que consiste en minimizar la distorsión para un rate determinado.

La combinación de MCTF y J2K ha sido propuesta en trabajos anteriores. Secker et al. se basan en esto para crear LIMAT [55], pero no se proponen algoritmos de rate-control o rate-allocation. En ellos la información de movimiento se coloca simplemente en primer lugar, seguida de los datos de texturas.

El rate-allocation se puede realizar durante la compresión o tras la misma. En la bibliografía sobre RA se encuentran algunas ideas interesantes para llevar a cabo durante la compresión.

Un enfoque similar al de LIMAT fue diseñado en [1] por André et al. y extendido en [12] por Cagnazzo et al. Además Ferroukhi et al. en [30], han propuesto recientemente un códec similar basado en la segunda generación de J2K. Los autores proponen un rate-allocation óptimo entre los datos de movimiento y texturas basado en el RDO (Rate Distortion Optimization, en español optimización del rate y la distorsión) el cual se inspira en la optimización de Lagrange, considerando que las distorsiones son aditivas, algo que puede ser sub-óptimo en aquellos casos en los que MCTF no es ortogonal.

En [58], se propone un algoritmo de rate-control para determinar la contribución de cada subbanda temporal para la codificación de vídeo de una sola capa. Este problema se ha resuelto para una tasa de bits objetivo (proporcionada por el usuario) controlando las diferentes fuentes de distorsión generadas durante el proceso de compresión y utilizando Lagrangian Optimization para encontrar la configuración óptima.

En [6] Barbarien et al. antes de utilizar la 2D-DWT, todos los coeficientes de residuos resultantes de la etapa MCTF se multiplican por un factor de escala para aproximar la MCTF a una transformación unitaria (preservando la energía). Así en el caso de la codificación de vídeo escalable en calidad, el problema es ligeramente diferente porque el code-stream está embebido y lo único que tenemos es un conjunto de puntos de truncamiento, no un bit-rate objetivo. En este contexto, el RA determina la composición de la información de un conjunto de puntos de truncado óptimos de la curva R/D, considerando un conjunto de bit-rates o un conjunto de distorsiones.

En [14], Cohen et al. proponen dos códecs J2K basados en ME (Motion Estimation, en español estimación de movimiento). El primero es un códec de pirámide 2D con un paso MCTF en cada nivel espacial y una estructura de codificación de bucle cerrado, similar a H.264/SVC [52] y HEVC [57]. El segundo códec es de bucle abierto, similar a MCJ2K, pero los autores no abordan el problema de rate-allocation entre las subbandas temporales y la información de movimiento.

Una alternativa interesante fue propuesta en [7] por Bilgin et al. donde se puede especificar un conjunto de pendientes R/D, una por cada capa de calidad, que incluya aquellas capas cuyas pendientes (en el dominio R/D) son mayores o iguales que la pendiente R/D deseada. Estos enfoques son óptimos sólo en los escenarios de transformación ortogonal, condición que es difícil de satisfacer (como se mostrará en los resultados experimentales) cuando se utilizan técnicas de estimación y compensación de movimiento.

Como se ha mencionado anteriormente, el rate-allocation puede realizarse en tiempo de descompresión. Y es en este caso, cuando puede ser implementado por el emisor (servidor), los receptores (clientes) o ambos. Algunos trabajos interesantes sobre rate-allocation en el lado servidor se describen a continuación.

FAST (Rate Allocation Through Steepest Descent) propuesto por Aulí-Llinàs et al. en [4] y mejorado por Jiménez-Rodríguez et al. en [26], es un algoritmo de RA dirigido por el emisor para las secuencias MJ2K. El método introducido en este trabajo selecciona una solución inicial (posiblemente con bajo rendimiento R/D), y la mejora de forma iterativa hasta que se agota el tiempo o el algoritmo termina la ejecución habiendo comprobado toda la casuística. Los resultados experimentales sugieren que FAST habitualmente logra soluciones cercanas al óptimo mientras emplea muy pocos recursos computacionales.

Otra interesante propuesta de RA dirigida por el servidor de MJ2K/MCJ2K fue introducida por Naman et al., la cual usa Conditional Replenishment (CR) [41] y Compensación de Movimiento (MC) [43]. En las propuestas de Naman, un servidor envía aquellos paquetes J2K relacionados con las regiones que los clientes deben refrescar para optimizar la calidad del vídeo después de considerar las restricciones de ancho de banda. Estas soluciones no son totalmente compatibles con J2K en el lado del servidor (un requisito en los servicios estándar de JPIP) porque se debe utilizar algún tipo de lógica no estándar de J2K.

En otros estudios también se han propuesto soluciones de RA impulsadas por el cliente.

En DASH [62], los clientes solicitan GOP a GOP aquellos segmentos del codestream que maximizan la QoE (Quality of Service, en español la calidad del servicio) del usuario, a través principalmente del nivel de llenado del buffer del cliente.

En [34], Mehrotra et al. proponen una mejora del enfoque anterior en el que los clientes utilizan la información R/D del vídeo para seleccionar (teniendo en cuenta la latencia de inicio deseada, el tamaño del buffer y la capacidad de red estimada) el número óptimo de capas de calidad (en el caso de H.264/SVC), o qué versión de calidad de cada GOP (en el caso de H.264/AVC) se transmitirá.

Nótese que la compatibilidad con JPIP (JPEG2000 Interactive Protocol) no se estudia en estos trabajos.

1.4 Streaming de vídeo escalable

La IPTV (Internet Protocol Television, en español televisión vía IP) ha despegado en los últimos años y se está extendiendo cada vez más. Hoy en día, la mayoría de los servicios de IPTV ya ofrecen tanto servicios de TV como de VoD (Video on Demand, en español vídeo bajo demanda).

1.4.1 Streaming de vídeo escalable vs. no escalable

Los servicios de streaming IPTV, especialmente aquellos en los que el vídeo se visualiza sin problemas mientras se transmite, necesitan cumplir los requisitos de heterogeneidad de las redes convergentes y la diversidad de terminales de los usuarios. Cuando los operadores de difusión de red entregan canales de TV a clientes heterogéneos, deben tener en cuenta la gran diversidad de requisitos del usuario final.

El contenido de vídeo puede ser entregado a los clientes, según dos estrategias básicas según Avramova et al. en [5]: streaming de vídeo a velocidad única o de velocidad variable.

En el enfoque de **vídeo no escalable**, el emisor transmite a una velocidad de bits fija a todos los receptores sin tener en cuenta (1) el ancho de banda disponible en cada momento de la conexión del cliente y (2) el perfil del terminal particular o necesidades de cada cliente. Estas necesidades comprenden un amplio abanico de posibilidades, como los formatos de vídeo soportados, la potencia de procesamiento, la memoria disponible, el modo de ahorro de energía, la disponibilidad de soporte de red de calidad de servicio, etc.

Se trata de la más popular. Usa vídeos no escalables entre los que se dispone de varias versiones (en términos de formato, calidad, codificación, etc., y por tanto de bit-rate) del mismo contenido de vídeo, permitiendo a los receptores decidir qué versión se ajusta a su perfil de comunicación mediante la conmutación entre múltiples code-stream distintos.

El streaming de **vídeo escalable** es más flexible y puede hacer un uso más eficiente de los recursos de la red. El vídeo se codifica en una capa base y varias capas adicionales de mejora, como propone Narroschke et al. en [44, 59]. Todas estas capas

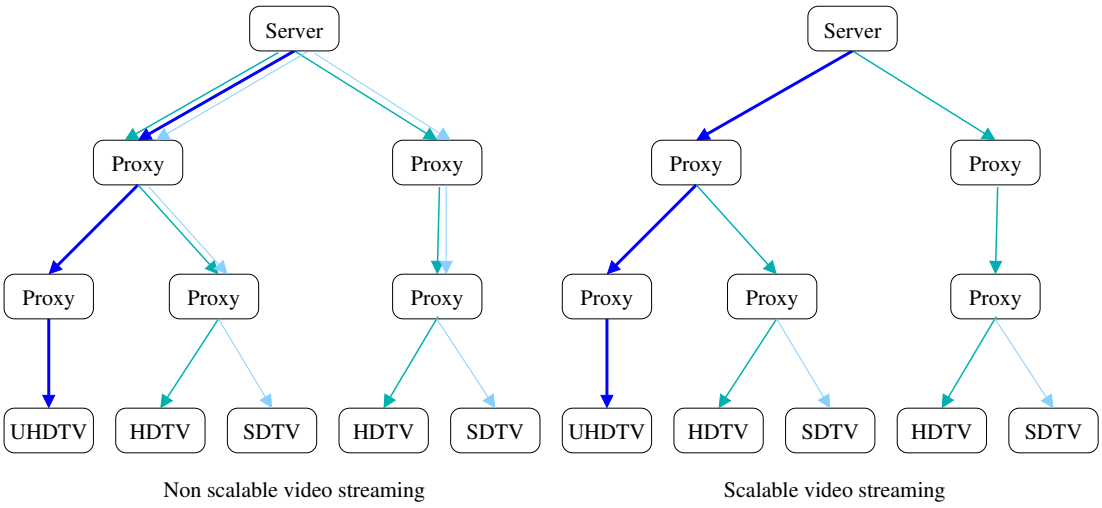


Figura 1.16: Streaming de vídeo escalable vs. no escalable.

de calidad están correlacionadas entre sí, de tal forma que para descodificar la capa n-ésima, se requiere la capa base y todas las capas hasta el nivel $(n-1)$-ésima. Por lo tanto, el code-stream de vídeo escalable resultante debería organizarse como una sucesión de capas acumulativas.

En la Fig. 1.16 se muestra un escenario típico donde se manifiestan las diferencias entre la transmisión de vídeo no escalable y escalable. En este ejemplo, un servidor transmite un vídeo escalable (en resolución) a varios clientes que demandan diferentes versiones del mismo utilizando un conjunto de proxies intermedios.

En el escenario en el que un servidor transmite vídeo no escalable, los usuarios finales demandan una copia del vídeo a sus proxies con la resolución deseada, desde los formatos de TV de ultra alta definición (UHDTV) hasta vídeo de definición estándar (SDTV). Las peticiones de vídeo se reenvían al servidor de vídeo, donde un conjunto de code-streams no escalables (que contienen digamos, la misma película) se entregan a los proxies para su distribución a los usuarios correspondientes. Observe que si cada code-stream está referenciado por una URL diferente, los usuarios pueden seleccionar la versión del vídeo que desean recibir.

En el escenario de difusión de vídeo escalable, sólo se envía un code-stream de vídeo escalable desde el servidor a los proxies, generando menos redundancia en la

transmisión. De nuevo, si cada capa del vídeo escalable tiene una URL diferente, los receptores pueden controlar qué resolución desean reproducir, siguiendo la regla de dependencia entre capas descrita anteriormente. Así, al solicitar un vídeo de un nivel de resolución determinado N, el receptor exigiría el nivel de resolución N-ésimo y todos los niveles de resolución inferiores, utilizando las URLs adecuadas. Esto llevaría a un mejor uso de los recursos de la red (ancho de banda), reduciendo la congestión que puede aparecer en los enlaces de la red del servidor (como sí sucede en el enfoque de transmisión de vídeo no escalable en la Fig. 1.16). Otra ventaja es que se descarga de trabajo al servidor, ya que la mayoría de las solicitudes de los clientes serían atendidas por proxies intermedios.

Existen alternativas para la transmisión de vídeo escalable, que aunque no tienen todas sus ventajas, sí son capaces de adaptarse a los requisitos del usuario:

1. MDC: Multiple Description Video Coding (en español codificación de vídeo con descripción múltiple) permite entregar contenido de vídeo en un enfoque de bit-rate variable. La diferencia con respecto a SVC es que el vídeo se codifica en varias versiones, denominadas descripciones, que pueden descodificarse de forma independiente para reconstruir el vídeo original. A medida que se disponga de más descripciones en el proceso de descodificación, la calidad del vídeo reconstruido será mayor. Por lo tanto, este enfoque es interesante ya que con una única descripción podemos reconstruir al menos una versión de baja calidad del vídeo que no será posible con SVC, cuando la capa base o al menos una capa de mejora por debajo de la capa de escalabilidad solicitada no esté disponible.

   Sin embargo para conseguir que cada capa se pueda reconstruir independientemente de las demás, la mayoría de los esquemas MDC sufren una redundancia significativa entre las diferentes descripciones y, en general, se consigue un rendimiento de R/D inferior al obtenido por SVC. En [18], Goyal et al. realizan una interesante introducción a la codificación de vídeo MDC.

2. Transcodificación: La transcodificación, presentada en [64] por Vetro et al. es otra tecnología que se puede utilizar para adaptar un code-stream de vídeo previamente no escalable a las capacidades del dispositivo del usuario final (ancho de banda, frecuencia de imágenes/segundo o resolución espacial). La desventaja

de esta técnica es que introduce una pequeña pérdida de calidad en los vídeos transcodificados en comparación con la compresión tradicional no escalable.

La complejidad de la transcodificación suele ser mucho mayor que la descodificación de vídeo escalable. Por lo tanto, exigir capacidades de transcodificación a todos los proxies intermedios para transmitir el vídeo a un conjunto de usuarios heterogéneos requeriría soluciones de hardware más complejas con sus correspondientes costes, comprometiendo la escalabilidad global del sistema.

En la actualidad, la mayoría de los servicios de transmisión de vídeo basados en IP, como YouTube, utilizan una transmisión no escalable para acomodar los diferentes requisitos de ancho de banda y resolución de los usuarios. El streaming no escalable es también la solución actual en los sistemas dedicados de DVB (Digital Video Broadcasting, en español difusión de vídeo digital). La razón principal para elegir el streaming no escalable es que los receptores no necesitan cambiar la infraestructura de descodificación de vídeo actual.

Por otro lado, aunque la entrega de vídeo escalable no es soportada por la gran mayoría de los descodificadores de vídeo, tiene varias ventajas si se compara con el streaming no escalable, especialmente en escenarios de ancho de banda variable. Las siguientes Secciones están dedicadas a describir cada una de estas ventajas. Nótese que aunque este trabajo tiene un énfasis especial en un enfoque de escalabilidad de calidad de grano fino, la mayoría de las ideas que se presentan aquí pueden ser extrapoladas a otros contextos en los que también se exige una multiresolución temporal o espacial.

1.4.1.1 **Coste de almacenamiento**

Una primera ventaja del uso de vídeo escalable consiste en el ahorro de recursos de almacenamiento en el lado del servidor porque, a pesar del número de resoluciones disponibles del vídeo, sólo se almacena una copia de cada vídeo (dividido en capas). En la versión de vídeo no escalable, se almacena una versión diferente para cada resolución, lo que impone importantes restricciones de memoria en grandes repositorios de vídeo. Este ahorro de almacenamiento también se produce en todos los servidores intermedios. Esto significa que, dado el conjunto de recursos de almacenamiento finitos, tanto los servidores como los proxies podrán tratar un mayor número de vídeos.

1.4.1.2 Uso del ancho de banda

En general, la distribución de vídeo escalable es menos redundante que la de vídeo no escalable equivalente, por lo que en términos de recursos de red, el vídeo escalable es más eficiente. Esto es evidente en varias tecnologías de difusión como, por ejemplo, la multidifusión (broadcasting) basada en IP y los sistemas P2P (Peer-to-Peer).

Cuando se utiliza la transmisión de vídeo escalable mediante IP, se puede realizar un ruteo que tenga en cuenta el contenido en los proxies estándar con la ayuda de protocolos como el RSVP (Protocolo de reserva de recursos) [69] (escrito por Zhang et al.). Así por ejemplo, cuando un vídeo se reenvía a través de un árbol de multidifusión, un proxy que recibe todas las capas podría transmitir los datos a cualquier otro, independientemente de la resolución que soliciten. Si queremos emular este comportamiento utilizando streaming no escalable, el proxy debería recibir el conjunto completo de vídeos no escalables, con la sobrecarga en bit-rate que esto representa [70] (escrito por Zou et al.).

Además, la sobrecarga producida por el protocolo de reserva es potencialmente mayor en el escenario de streaming no escalable. Para mostrar esta sobrecarga, suponga que un usuario final demanda una resolución de vídeo en un área de red donde el resto de usuarios están recuperando una versión de mayor resolución del mismo vídeo. Si se utiliza la transmisión no escalable, la solicitud de reserva del usuario tiene que viajar hacia arriba en el árbol de multidifusión hasta que coincida con un proxy que esté enviando la resolución solicitada, y después de eso crear la nueva rama de multidifusión.

En el caso de vídeo escalable, esta sobrecarga se evita ya que probablemente el primer proxy estará enviando una versión de mayor resolución del vídeo, por lo que puede procesar directamente las peticiones del usuario entregando la resolución solicitada.

En los sistemas de difusión en tiempo real basados en P2P, los peers (nodos) pueden extrapolar la idea de un streaming de vídeo escalable para aumentar las posibilidades al compartir datos entre ellos, lo cual es esencial para el rendimiento óptimo de la red [47]. Por ejemplo, un peer que esté recibiendo la versión de mayor resolución del vídeo (es decir, todas las capas) también puede compartir datos con el resto de peers, independientemente de la resolución demandada. Esta ventaja de ofrecer vídeo escalable también está disponible en los sistemas estándar servidor-proxy-cliente.

1.4.1.3 Streaming con rate-control

Aunque este trabajo se ha centrado especialmente en la transmisión de vídeo escalable en calidad, la mayoría de los códecs de vídeo escalables incorporan además escalabilidad temporal y espacial. Gracias a esto, el número de versiones que realmente tiene un vídeo escalable es igual a:

$$\begin{gathered}\text{el número de resoluciones espaciales} \times \\ \text{número de resoluciones temporales} \times \\ \text{número de niveles de calidad.}\end{gathered}$$

Aunque teóricamente es posible diseñar un sistema de streaming no escalable con muchas versiones del vídeo, en la práctica esto no es factible porque la redundancia global sería excesiva. Teniendo en cuenta esta diferencia clave, está claro que en aquellas situaciones en las que existe un ancho de banda variable en el tiempo, muy común en la Internet pública, el rate-control se puede hacer con mayor precisión con el vídeo escalable que con el no escalable.

Observe también que el uso de vídeo escalable no implica cambiar el comportamiento estándar de los proxies web. Como se ha mencionado anteriormente, se puede acceder a cada capa de un vídeo escalable utilizando una URL diferente.

1.4.1.4 Coste de la resistencia a errores

En las redes propensas a errores como los enlaces inalámbricos, la protección contra errores puede ser conveniente. En este contexto, las técnicas FEC (Forward Error Correction, en español corrección de errores de envío) podrían aplicarse con una mínima redundancia utilizando una protección de errores desigual, introducida por Maani et al. en [31]. La idea consiste en proteger más las capas más importantes del vídeo. Ésta es la capa base y en menor medida progresivamente cada una de las capas adicionales.

Con un streaming no escalable se podría aplicar una protección de errores a cada versión del vídeo, pero la redundancia introducida en todos los streams es superior a la requerida para proteger adecuadamente la versión de vídeo escalable. Esto se debe a que la cantidad de datos a proteger, en multitud de versiones del vídeo no escalable es mucho mayor que en el concepto de un único vídeo escalable. Además no se podría seleccionar a priori qué versiones del vídeo son más o menos importantes para ser protegidas en distinta medida.

1.4.2 Protocolo interactivo de JPEG2000

El estándar se basa en JPIP (JPEG2000 Interactive Protocol, en español protocolo interactivo JPEG2000) [23] para transmitir un code-stream J2K en sistemas cliente/servidor ofreciendo un alto grado de escalabilidad (espacial, temporal y en calidad). Esta característica hace que J2K (y su extensión MJ2K) sea especialmente adecuado para la gestión de repositorios de vídeo y para la implementación de servicios interactivos de streaming de imagen/vídeo, tal como expone Bilgin et al. en [7].

En particular, JPIP ha demostrado ser muy eficaz para la visualización de datos de imágenes del Sol a escala de petabytes (Helioviewer Project), que permite el streaming interactivo de imágenes y secuencias de imágenes [50], permitiendo a los investigadores y al público en general explorar datos de imágenes dependientes del tiempo procedentes de diferentes observatorios espaciales, hacer zoom interactivo en zonas de interés (WOI) y reproducir secuencias de imágenes de alta resolución a varias cadencias (imágenes/segundo). Cuando un cliente JPIP solicita a un servidor una WOI de una secuencia de imágenes, el servidor usando estrictamente la funcionalidad JPIP, envía al cliente la progresión de paquetes adecuada para el streaming solicitado.

Capítulo

2

MCJ2K

Este capítulo presenta nuestra propuesta. Más concretamente, en la Sec. 2.1 se describe la codificación del códec de vídeo MCJ2K (Motion Compensated JPEG2000), una combinación de Motion JPEG2000 con compensación de movimiento y técnicas de rate-allocation. El RA (Rate Allocation) realizado durante la codificación mejora el rendimiento R/D de las reconstrucciones, ya sea del code-stream descodificado completamente o sólo parte de él. En la Sec. 2.2 se proponen soluciones de RA para MCJ2K en tiempo de post-compresión que aumentan el rendimiento R/D de la descodificación del code-stream truncado. En la Sec. 2.3 se detalla una representación del code-stream resultante de la codificación y el RA en post-compresión realizado por MCJ2K. Finalmente, en la Sec. 2.4 se muestran las posibilidades de la descodificación progresiva de un code-stream MCJ2K.

2.1 Codificación de MCJ2K

MCJ2K es nuestra propuesta de códec de vídeo basado en los tres principios para lograr un rendimiento R/D competitivo: (1) codificación de vídeo escalable, para adaptarse a clientes heterogéneos, (2) explotación de la correlación temporal entre las imágenes, lo que reduce el bit-rate necesario para el streaming de vídeo, y (3) uso de técnicas de RA para maximizar la utilidad de los datos transmitidos, especialmente cuando el ancho de banda sólo permite el streaming del code-stream truncado [14].

La Fig. 2.1 muestra la arquitectura de nuestra implementación del códec de vídeo MCJ2K. La secuencia original esta representada por V, donde $V^{[Q-1]}$ es una aproximación progresiva de V, reconstruida con MCJ2K utilizando Q capas de calidad. MCTF^T transforma V en una colección de $T+1$ subbandas temporales de texturas

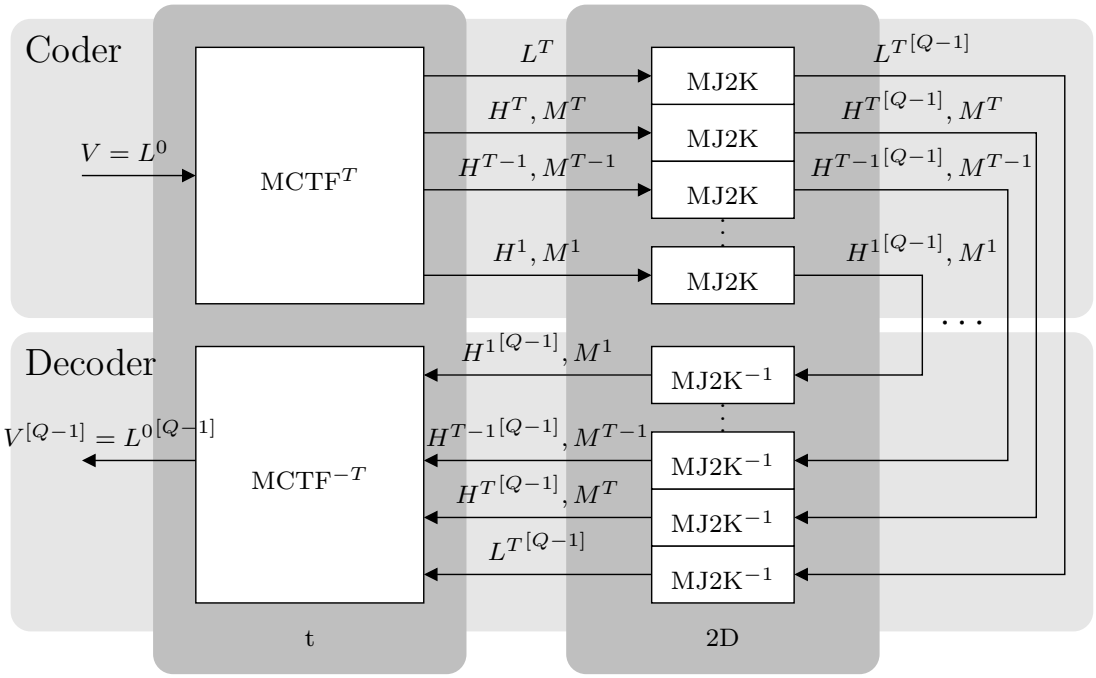


Figura 2.1: Arquitectura del códec MCJ2K.

$\{L^T, \{H^t; 1 \leq t \leq T\}\}$, y T "subbandas" de movimiento $\{M^t; 1 \leq t \leq T\}$.

MCJ2K es una estructura de lazo abierto según "t+2D". La "t" se corresponde a un esquema MCTF de T-niveles (MCTF^T), basado en una transformada 1/3 lineal 1D-DWT. El "2D" hace referencia a una transformada wavelet 2D-DWT, realizada por el códec estándar J2K. Por tanto, MCTF^T explota la redundancia temporal y 2D-DWT, incluida como parte de los compresores J2K, la redundancia espacial.

MCJ2K codifica una secuencia de imágenes a través de las siguientes etapas:

1. MCTF descorrelaciona temporalmente las imágenes del vídeo, distribuyéndolas en un conjunto de subbandas temporales y campos de movimiento. En función del grado de detección de la correlación temporal del vídeo, las imágenes son codificadas en dichas subbandas como intra o bidireccionalmente predichas, (véase la Sec. 2.1.1). Se pueden encontrar más detalles sobre cómo se puede implementar MCTF en [53], y en nuestra implementación publicada en GitHub (véase la Sec. 3.1).

2. Cada subbanda temporal se comprime (véase la Sec. 2.1.2) usando el códec

MJ2K dividiendo cada coef (imagen en una subbanda) en capas de calidad, generando una colección de SLs (subband-layers, en español capas de subbanda). Estas SLs suponen los puntos óptimos de truncado de cada code-stream MJ2K (cada subbanda). Considerando el code-stream completo (conjunto de subbandas) y su proceso de su descodificación, se ha desarrollado una técnica de RA, llevada a cabo durante la codificación, que maximiza la relación R/D de las reconstrucciones mediante la selección de los RD-slopes para la codificación de las subbandas.

La compatibilidad de MCJ2K en los actuales entornos de streaming cliente/servidor viene dada por su codificación basada en el estándar MJ2K. Por lo tanto, un code-stream MCJ2K es totalmente compatible con cualquier servidor JPIP estándar, ya que todo el code-stream MCJ2K puede ser almacenado en archivos MJ2K estándar. El uso de MCJ2K en lugar de J2K en los sistemas de streaming JPIP podrá mejorar el rendimiento R/D en la transmisión de secuencias de imágenes.

En los clientes, para reproducir un bit-stream MCJ2K, cada subbanda temporal (un code-stream MJ2K) debe ser descomprimido (lo cual también es una operación estándar). Después se ejecuta una etapa MCTF inversa para renderizar el vídeo.

Adoptar MCJ2K en entornos cliente/servidor implica la recompresión de las secuencias de vídeo con este códec, pero no necesita de modificaciones en servidores JPIP. Sólo la parte del cliente JPIP necesita implementar la lógica necesaria para MCJ2K.

2.1.1 Etapa MCTF en MCJ2K

En la primera etapa, MCTF aplicado a nivel de GOP, produce una representación temporal multiresolución de las imágenes y disminuye significativamente la redundancia temporal. Se trata de una transformación que toma como entrada una secuencia de imágenes y produce una secuencia de coefs, agrupados en una colección de subbandas temporales.

Como puede verse en la Fig. 1.9, MCTF produce un conjunto de subbandas temporales, donde cada una es una secuencia de imágenes. Se crearán tantas subbandas temporales como indique el parámetro de número de TRLs (Temporal Resolution Levels, en español número de resoluciones temporales) indicado en tiempo de compresión.

Las subbandas temporales contienen entonces información sobre las texturas. Cada subbanda está compuesta por imágenes I y/o residuos B, que suelen comprimirse con pérdida, en un número determinado (Q) de capas de calidad. Cada capa de calidad transmitida durante un streaming proporciona una información más precisa para reconstruir las imágenes, por lo que la distorsión disminuye. Hay un límite en el número de capas en cada subbanda suficientemente alto (65536), pero un número demasiado alto causa un coste ya que cada capa genera nuevos paquetes J2K. Además, la información de una subbanda quizá no pueda ser dividida en más capas, resultando capas vacías. Las capas vacías son creadas sin información útil, sólo contienen su cabecera, siendo una capa que aumenta el bit-rate, sin disminuir la distorsión de la reconstrucción.

En la Fig. 1.9 se presentan dos tipos de subbandas, L y H. En el code-stream habrá una subbanda L y tantas H como número de TRLs. A continuación se describen dichas subbandas.

1. *Subbanda L (Low):* Esta subbanda contiene las texturas del último coefs de tipo intra (I). Por lo tanto, su codificación es independiente del resto de las imágenes de la secuencia.

2. *Subbandas H (High):* Contienen los coefs producidos por el algoritmo MCTF. Son la diferencia entre las imágenes predichas y las originales. Por lo tanto, cuanto mejor sea la predicción, menor será la cantidad de datos que contengan estos coefs. A cada uno de estos coefs se denominará residuo (coef tipo B).

   Puede ocurrir que un residuo tenga un bit-rate mayor que el de la misma imagen si fuera codificada como I, por lo que en este caso, al no ser provechoso el uso de la compensación de movimiento, la imagen se codifica como I, no como B. A este proceso se denomina poda (véase la Fig. 2.2). Así, en caso de no encontrar suficiente correlación temporal, se eliminan las referencias en los MVs y el coef B se sustituye por su codificación I.

2.1.1.1 **Predicción del movimiento**

Las "subbandas" de movimiento son un conjunto de MVs, calculados en el proceso MCTF y asociados a cada coef predicho. Los vectores de movimiento indican la posi-

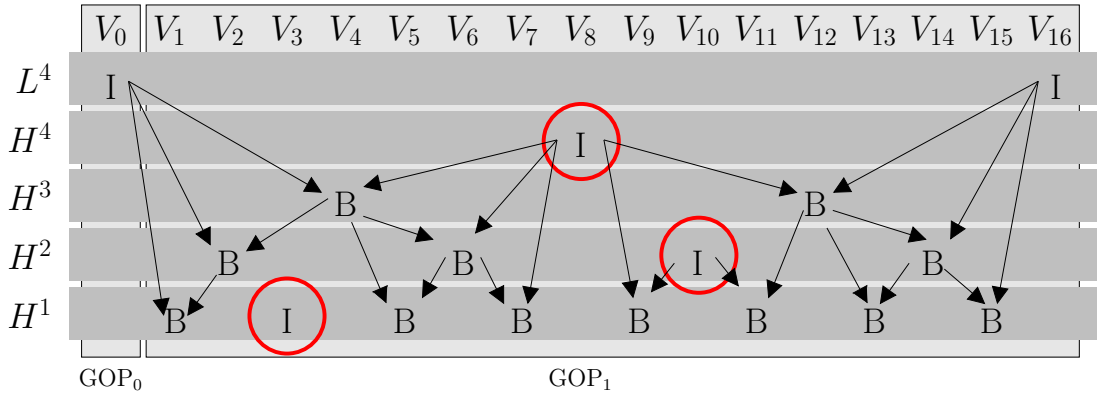


Figura 2.2: Poda en MCTF.

ción relativa de las texturas de un coef de tipo B con respecto a sus referencias (otras dos imágenes de resoluciones temporales superiores), según el esquema MCTF.

MCJ2K implementa una búsqueda de macro-bloques entre imágenes que puede ser exhaustiva y logarítmica [19], y a nivel de píxel o sub-píxel [56]. Dicha búsqueda de macro-bloques entre imágenes se realiza mediante un algoritmo de estimación de movimiento bidireccional [66].

La Fig. 2.3 muestra un posible proceso de eliminación de redundancia temporal en una secuencia de tres imágenes. El proceso de estimación de movimiento divide las imágenes en macro-bloques y busca la posición de cada uno de ellos en las imágenes vecinas (no tienen por qué ser las imágenes adyacentes), o al menos el macro-bloque más similar. Entonces se almacena la posición relativa del macro-bloque respecto a las otras imágenes de referencia mediante vectores (llamados MVs). Hay dos MVs para cada macro-bloque de cada imagen predicha, uno relativo a la imagen de referencia posterior y otro a la anterior.

Los principales parámetros de búsqueda son (1) el tamaño de los macro-bloques, el cual es constante (dentro de un coef) y (2) el área de búsqueda.

1. *Tamaño de macro-bloque*: El mejor tamaño para los macro-bloques es aquel que coincide con el de los objetos (texturas en las imágenes) que se pretenden rastrear. Por un lado, los bloques de mayor tamaño son menos sensibles al ruido, en caso de encontrar similitudes altas en varias posiciones. Por otro lado, un

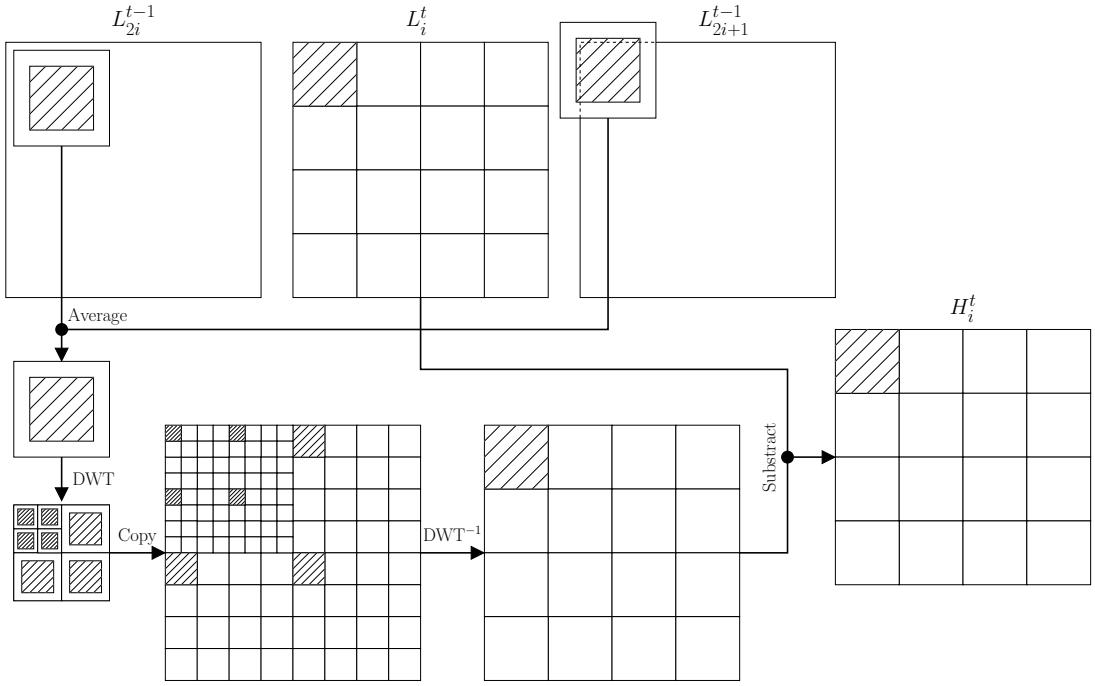


Figura 2.3: Flujo de datos en la predicción de movimiento.

tamaño reducido permite encontrar mejores similitudes ya que sólo tienen que coincidir pequeñas partes entre las imágenes.

2. *Área de búsqueda*: Expresada en número de píxeles, está determinada por la posición del macro-bloque en la imagen actual, más dicha distancia "área de búsqueda" alrededor de él. Se usa para disminuir el tiempo de cómputo para la búsqueda de la mayor similitud, restringiendo la búsqueda a un área al rededor del macro-bloque de referencia. Su tamaño se especifica como parámetro en la codificación.

La energía presente en las imágenes, en este caso los macro-bloques, tiene una frecuencia espacial muy baja, que además varía lentamente según transcurre la secuencia. Por tanto, al aplicar una transformada se concentra la mayor parte de la energía en unos pocos coeficientes (en las bajas frecuencias), provocando que en el resto de subbandas espaciales la mayoría de los coeficientes sean cero o casi cero. Como la distribución de los coeficientes no es uniforme, la compresión consiste en llevar a cero los coeficientes

que ya están cerca de cero y cuantificando los restantes.

La transformada en sí, no reduce el rate necesario para la representación del macro-bloque, pero permite distribuir la información de forma que pueda ser cuantificada, afectando a los coeficientes menos importantes en mayor medida y más importantes en menor, de forma progresiva según el nivel de cuantificación.

Las pérdidas por la cuantificación de los macro-bloques unido a que la predicción del movimiento no tiene por qué ser exacta produce el "efecto de bloque" (en inglés blocking). El blocking consiste en la visualización de pequeñas discontinuidades entre los macro-bloques adyacentes una vez se reconstruye la imagen. Este efecto es menor al haber cuantificado los coeficientes según la distribución realizada por la transformada.

Finalmente para la generación del residuo (diferencia entre la imagen predicha y la original), es necesario primero volver al dominio de la imagen, realizando la transformada inversa.

Estas operaciones de estimación/compensación de movimiento (ME/MC) se realizan a la máxima resolución espacial de la secuencia. Esta decisión de diseño, que es conveniente para una visualización del vídeo de resolución progresiva, implica que el proceso de compensación de movimiento inverso debe realizarse también a la máxima resolución para evitar un error de deriva (shift-error) [37], cuando se descodifica una resolución reducida de las imágenes. Obviamente en este caso, el descodificador aumenta los requisitos de computación, pero esto no aumenta significativamente el uso de la memoria, a menos que se procesen todos los macro-bloques en paralelo. Como ventaja, la calidad de las reconstrucciones es mayor que en el caso de que la ME/MC se realice a una resolución menor, porque la información de movimiento se utiliza siempre con la precisión utilizada en la compresión, que puede ser sub-píxel.

Los datos de movimiento son descorrelacionados temporal y espacialmente, y comprimidos sin pérdida con el códec J2K, como una secuencia de imágenes de una sola capa ($Q = 1$) de 4-componentes (2 vectores por macro-bloque). Usualmente, el uso de información aproximada en los MVs genera artefactos severos en las imágenes reconstruidas e incrementa el grado de no ortogonalidad del MCTF.

El proceso de descorrelación utiliza un algoritmo en el que, cuando no se reciben datos de movimiento, el proceso MCTF inverso supone que los MVs indicarían un

movimiento lineal. Así en el proceso de transmisión, cuando el descodificador conoce M^T, pero no M^{T-1}, supone que los MVs de M^{T-1} son la mitad del valor de M^T. Esta predicción lineal se utiliza para el resto de niveles de resolución temporal [3].

2.1.1.2 GOPs en MCJ2K

En MCJ2K los GOPs son abiertos y simétricos en cada resolución temporal. MCJ2K usa GOPs abiertos para minimizar el número de imágenes intra en cada GOP, y por tanto en el code-stream. Esto es interesante porque las imágenes intra suelen tener un rate significativamente mayor que las imágenes B dada una distorsión

En la etapa MCTF inversa de MCJ2K, se reconstruyen las subbandas temporales a partir de (1) las imágenes B del GOP actual y (2) la última del GOP anterior. MCJ2K podría usar GOPs cerrados de la misma forma que ahora usa abiertos, igualmente la descodificación se realizaría dependiendo de dos imágenes intra (una anterior y otra posterior). La diferencia en MCJ2K si usara GOPs cerrados sería que la imagen intra anterior, sería la primera del GOP actual (y no la última del GOP anterior). Por tanto, cada GOP tendría dos imágenes intra (la primera y la última), lo cual provoca dos problemas: (1) el número de imágenes intra aumentaría al doble, con el consiguiente aumento del rate, y (2) no tendría sentido tener dos imágenes de referencia consecutivas.

En MCJ2K a partir de estas imágenes intra de referencia, se genera un árbol de dependencias simétrico si la predicción de movimiento es exitosa. Si no lo fuera se realizará el proceso de poda, haciendo asimétrico dicho árbol (véase la Fig. 2.2).

2.1.2 Etapa MJ2K en MCJ2K

En la segunda etapa, el compresor de imagen J2K se utiliza para codificar los residuos que generó MCTF e incorporar escalabilidad espacial y en calidad. En otras palabras, los coefs son comprimidos con J2K, resultando en una colección de bit-streams J2K. Cada subbanda temporal se comprime en capas de calidad, generando una colección de SLs.

2.1.2.1 Rate allocation durante la compresión

Las técnicas presentadas en esta sección se llevan a cabo durante la codificación de las imágenes, es decir, durante la generación del code-stream, no durante su transmisión. Por lo tanto, estas técnicas no afectan a la arquitectura actual de los clientes o servidores. En general, durante la compresión, el rate-allocation (también llamado rate-control) determina la cantidad de información que se debe incluir en cada SL de cada subbanda del code-stream para optimizar la relación R/D de las reconstrucciones.

En este trabajo se propone un enfoque diferente a los descritos en la Sec. 1.3.6. En concreto, respecto de [58] en nuestra propuesta se usan múltiples capas de calidad por coef. Respecto de [7], nuestras soluciones de rate-allocation determinan también las capas de calidad con mayor pendiente R/D, pero lo hacen tras un proceso no ortogonal (la codificación de vídeo con ME/MC mediante nuestro MCTF), en [7] no se tiene esta problemática al tratarse de codificación de imagen, no de vídeo con movimiento compensado.

En este nuevo planteamiento los R/D-slopes para determinar la calidad de las capas al comprimir los residuos con J2K, dependen de la energía de la subbanda temporal. Así, ignorando cualquier posible efecto del comportamiento no ortogonal que provoque la etapa ME/MC en la etapa MCTF inversa, nuestra implementación de MCTF se aproxima a una transformación biortogonal y por lo tanto, cada subbanda $\{L^T, H^T, \cdots, H^1\}$ contribuye con una cantidad diferente de energía a la reconstrucción de la secuencia, y además es posible que los coefs no sean (dependiendo de la fase MC) independientes. Esto puede ser fácilmente verificado, comparando la energía que los diferentes coefs de cada subbanda temporal contribuyen a la reconstrucción de la secuencia. Usando dicha energía, definimos la atenuación

$$a_t = \frac{E(L_i^T)}{E(H_j^t)}, \tag{2.1}$$

donde $E(\cdot)$ representa la energía del correspondiente coef, L_i^T y H_j^t, con $0 < t \leq T$. Estas atenuaciones son empíricas, determinadas específicamente para la DWT de 1/3 ME implementada en nuestro códec (para una transformación diferente, se obtendrían otros valores).

En MCJ2K, las SLs de cada GOP comprimido están determinadas por un conjunto $\{\lambda_0^s, \cdots, \lambda_{Q-1}^s\}$ de RD-slopes, donde Q es el número total de SLs por subbanda y s

Tabla 2.1: Norma-L_2 (energía) de las funciones base de la MCTF para las subbandas temporales, expresadas como valores de atenuación.

MCTF^1		MCTF^2		MCTF^3		MCTF^4		MCTF^5		MCTF^6		MCTF^7	
H^t	a_t	H^t	a_t	H^t	a_t	H^t	a_t	H^t	a_t	H^t	a_t	H^t	a_t
H^1	1,246	H^2	1,250	H^3	1,160	H^4	1,088	H^5	1,046	H^6	1,023	H^7	1,012
		H^1	1,865	H^2	2,122	H^3	2,130	H^4	2,079	H^5	2,043	H^6	2,023
				H^1	3,167	H^2	3,888	H^3	4,061	H^4	4,063	H^5	4,039
						H^1	5,802	H^2	7,431	H^3	7,936	H^4	8,031
								H^1	11,089	H^2	14,522	H^3	15,688
										H^1	21,669	H^2	28,707
												H^1	42,835

la correspondiente subbanda temporal. Teniendo en cuenta que las SLs de cada GOP son los puntos de truncado óptimos de la subbanda temporal a la que pertenece, la calidad de los coefs de una subbanda puede considerarse constante si el code-stream se trunca en una SL [15]. En consecuencia, utilizando los mismos RD-slopes para cada coef de cada subbanda, si los procesos MCTF+J2K fueran ortogonales [6], Q puntos de truncado óptimos por subbanda estarían disponibles cuando se descodificaran el mismo número de SLs de cada subbanda.

La cantidad de energía que un coef debe contribuir al code-stream para aproximar la MCTF a una transformación ortonormal o unitaria (preservando la energía) se representa mediante factores de atenuación en la Tab. 2.1. Esta tabla se ha calculado teniendo en cuenta la ganancia de energía de las funciones base producidas por los filtros de síntesis que se corresponden con cada subbanda, y el número de niveles de MCTF [67]. Por ejemplo, en MCTF^2, un coef de la subbanda L^2 es $1{,}25$ veces más relevante en la reconstrucción del GOP que el coef de la subbanda H^2, y $1{,}865$ veces más importante que un coef de la subbanda H^1.

Estos factores de atenuación pueden utilizarse de dos maneras diferentes: (1) multiplicando cada (píxel de cada) coef por el factor correspondiente y cuantificando por un valor constante (este es el enfoque tradicional, usado por ejemplo en [6]), o (2) controlando los RD-slopes usados en la etapa MJ2K cuando se definen las SLs (este procedimiento se usa en nuestra propuesta).

Los RD-slopes usados por MCJ2K son definidos por

$$\lambda_0^{H^t} = \lambda_0^{L^T} + \frac{\Delta d}{J} a_t, \tag{2.2}$$

donde $\lambda_0^{L^T}$ es el RD-slope especificado por el usuario para establecer la calidad que será proporcionada por la SL base $L^{T^{(0)}}$ (lo que equivale a decir, la calidad de $V^{[0]}$ porque no se ha usado un filtro de síntesis paso bajo en nuestra implementación de MCTF), Δd y J son dos constantes que dependen de la implementación del códec J2K, y a_t es el factor de atenuación para la ganancia de la subbanda temporal H^t en relación con L^T. Por ejemplo, en Kakadu [27], $\Delta d = 256$ (RD-)slope-steps (un valor considerado adecuado para evitar la generación de paquetes J2K vacíos) y $J = \sqrt{2}$ (el incremento de ratio esperado en la longitud del code-stream cuando el RD-slope se decrementa en Δd).

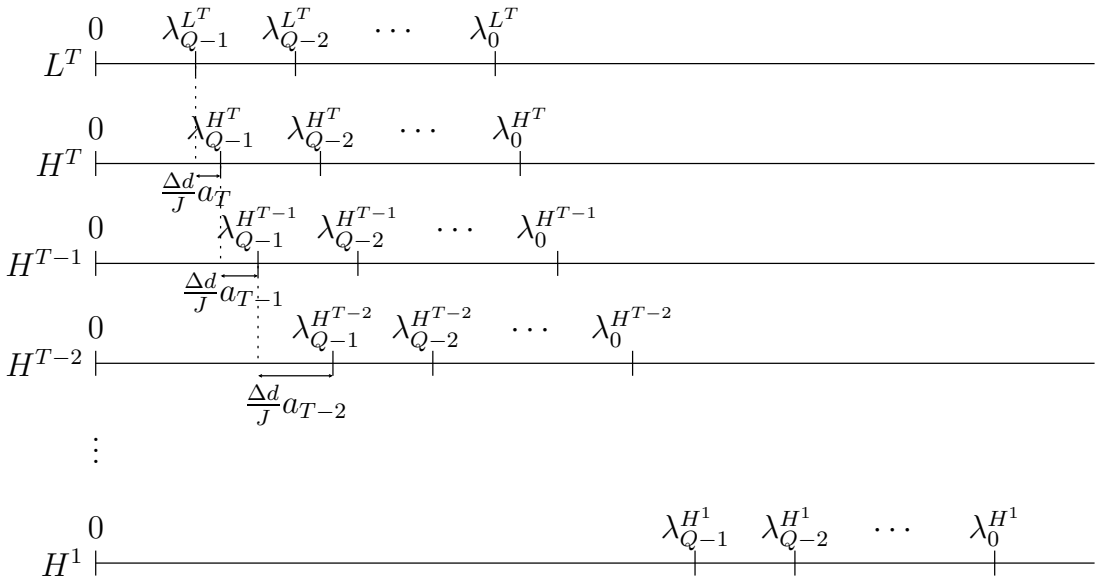


Figura 2.4: RD-slopes usados en las diferentes subbandas temporales para maximizar la calidad dado un bit-rate.

La Eq. 2.2 establece la relación entre la distorsión esperada, generada en cada subbanda temporal H^t en función de la primera pendiente $\lambda_0^{H^t}$ utilizada para esa subbanda, que depende de $\lambda_0^{L^T}$ y la atenuación para cada H^t. En MCJ2K, el resto de las SLs de calidad están definidas por

$$\lambda_q^s = \lambda_0^s + \frac{\lambda_0^s - \lambda_{Q-1}^s}{Q} q, \tag{2.3}$$

donde $s \in \{L^T, H^T, H^{T-1}, \cdots, H^1\}$, $1 \leq q < Q$, y λ_{Q-1}^s es la pendiente que determina la máxima calidad de s obtenida cuando se han descomprimido todas las SLs de s. En la Fig. 2.4 se muestran los RD-slopes, que son equidistantes en cada subbanda temporal y se encuentran atenuados respecto de $\lambda_0^{L^T}$.

En la etapa MJ2K, cada coef de cada subbanda de texturas está comprimido con dicho códec, produciendo un code-stream de longitud variable en capas. Supongamos que s_i^q es la capa de calidad q de la representación comprimida del coef s_i de la subbanda s, y que $s^{(q)}$ es la calidad (es decir, la disminución de la distorsión) que proporciona en la reconstrucción progresiva de s_i al usar la capa q. Asumiendo que la métrica de

distorsión es aditiva, definimos

$$s^{[q]} = \sum_{j=0}^{q} s^{(j)}, \tag{2.4}$$

como la calidad de la reconstrucción de la subbanda s usando q capas. Definimos la pendiente q-ésima RD de la subbanda s como

$$\lambda_q^s = \frac{s^{[q]} - s^{[q-1]}}{l(s^q)} = \frac{s^{(q)}}{l(s^q)}, \tag{2.5}$$

donde $l(s^q)$ representa la longitud (rate) de s^q.

Debido a la forma en que se eligen los RD-slopes en la etapa MJ2K, para dos coefs diferentes i y j de la subbanda s se verifica que

$$\lambda_q^{s_i} = \lambda_q^{s_j}, \forall q \in \{0, \cdots, Q-1\}. \tag{2.6}$$

La Eq. 2.6 tiene dos implicaciones: (1) en general, la longitud total del code-stream de cada coef será diferente (dependiendo de su contenido aunque estén cuantificados con igual valor de q), y (2) el rate-allocation es óptimo para cada SL [15].

2.2 Rate allocation en post-compresión

En escenarios con restricciones de ancho de banda, el término rate-control se utiliza para decidir qué partes del code-stream se descodifican y cuales no (para un rate dado). En otras palabras, el rate allocation tiene el objetivo de optimizar la reconstrucción del vídeo desde el punto de vista de la relación entre el rate y la distorsión.

En este trabajo se presentan dos algoritmos de rate-allocation, que realizan rate-control del code-stream a nivel de GOP. En concreto dichos algoritmos definen el orden en que las SLs deben ser descodificadas para optimizar la progresión en calidad. Las técnicas propuestas son: OSLA (Optimized subband Layers Allocation, en español asignación de SLs optimizado) y ESLA (Estimated-slope subband Layers Allocation, en español asignación de SLs con slope estimado).

Ambos algoritmos se ejecutan tras la etapa de codificación y antes de su transmisión, una sola vez por vídeo, independientemente del número de veces que se haga streaming del mismo. Además, no afectan al contenido del code-stream MCJ2K por lo

que sigue siendo compatible de igual forma con la arquitectura actual cliente/servidor. De hecho no es necesario hacer ninguna modificación en el lado del servidor.

Para realizar el RA proponemos encontrar un orden de las SLs de cada subbanda temporal, que satisfaga que las pendientes de la curva R/D entre los puntos de truncado óptimos sean lo más altas posibles. De esta forma, en la transmisión progresiva (una a una) de las SLs se obtendrá menor distorsión a igual rate.

Las SLs en un code-stream MCJ2K son

$$\begin{array}{llll} L^{T^0}, & H^{T^0}, & H^{T-1^0}, & \cdots, \quad H^{1^0}, \\ L^{T^1}, & H^{T^1}, & H^{T-1^1}, & \cdots, \quad H^{1^1}, \\ \vdots & \vdots & \vdots & \vdots \\ L^{T^{Q-1}}, & H^{T^{Q-1}}, & H^{T-1^{Q-1}}, & \cdots, \quad H^{1^{Q-1}}, \\ & M^T, & M^{T-1}, & \cdots, \quad M^1, \end{array} \tag{2.7}$$

y hay $Q(T+1)$ SLs en este conjunto, que es también el número de puntos de truncado óptimos de un code-stream MCJ2K.

El RA se realiza normalmente durante la descompresión. En la Parte 9, Sec. C.4.10 de la norma J2K [23], los clientes JPIP pueden solicitar imágenes J2K por capas de calidad. Esto se lleva a cabo en el códec J2K, simplemente transmitiendo las capas de calidad de una imagen, en orden decreciente de cuantificación. Sin embargo, en un códec que reduce la correlación temporal, por ejemplo MCJ2K, algunas imágenes dependen de otras para su reconstrucción y no puede aplicarse la misma metodología. Además, como se ha estudiado anteriormente por Sánchez et. al en [50], también es posible realizar una petición JPIP para un rango de imágenes.

Por lo tanto, por extensión el estándar JPIP también puede utilizarse para recuperar SLs de varios coefs utilizando una sola solicitud JPIP. Sirvan los siguientes ejemplos para ilustrar la petición JPIP respecto de una o varias imágenes de una secuencia:

1. En la petición de un sólo coef, por ejemplo, en la Fig. 2.7 la SL H^{2^1} tiene para cada GOP sólo un coef (H_0^2). Este coef tiene tres capas de calidad $\{H_0^{2^0}, H_0^{2^1}, H_0^{2^2}\}$, de las cuales $\{H_0^{2^0}, H_0^{2^1}\}$ serían las que solicita un cliente para recuperar la SL H^{2^1}.

2. En la petición de dos coefs, por ejemplo, en la Fig. 2.7 la SL L^{2^0} tiene dos coefs (L_0^2 y L_1^2). Éstos tienen tres capas de calidad por coef $\{L_0^{2^0}, L_0^{2^1}, L_0^{2^2}\}$ y

$\{L_1^{2^0}, L_1^{2^1}, L_1^{2^2}\}$ respectivamente. De las cuales $\{L_0^{2^0}, L_1^{2^0}\}$ serían las que solicita un cliente para recuperar la SL L^{2^0}.

En el momento de la descodificación, el orden en el que se recuperan las SLs del servidor JPIP debería minimizar la curva R/D para una tasa de bits objetivo. Para esta tarea proponemos los dos enfoques siguientes.

2.2.1 Optimized SL Allocation (OSLA)

Empezando la transmisión por la SL L^{T^0}, el orden óptimo de las restantes SLs de un GOP puede determinarse aplicando la Eq. 2.5 para cada SL factible de ser enviada, como continuación a la actual progresión de SLs en el streaming. Una vez calculadas las pendientes de todas las SLs que pueden ser enviadas como próximo paso en la transmisión, se selecciona la de mayor pendiente para ser descodificada. Así OSLA selecciona una a una, la mejor SL que se puede adicionar al code-stream.

Sirva como ejemplo la siguiente progresión: después de transmitirse L^{T^0} (la cual siempre contribuye a la calidad de la reconstrucción más que cualquier otra SL), se presentan varias alternativas $\{M^T, M^{T-1}, \cdots, M^1, H^{T^0}, H^{T-1^0}, \cdots, H^{1^0}, L^{T^1}\}$ que deben ser comprobadas para determinar la próxima SL que provoque la mayor pendiente posible en la curva R/D. Continuando con el ejemplo propuesto, y suponiendo que $\lambda^{M^T} > \lambda^{M^t}, \forall\, t \in \{T-1, \cdots, 1\}$, consideremos tres escenarios distintos:

1. Si $\lambda^{M^T} > \lambda_0^{H^t}, \forall\ t \in \{T, \cdots, 1\}$, la siguiente SL a descodificar debería ser M^T y el siguiente conjunto de alternativas sería $\{M^{T-1}, H^{T^0}, H^{T-1^0}, \cdots, H^{1^0}, L^{T^1}\}$.

2. De lo contrario, si $\lambda_0^{H^T} > \lambda^{M^t}$ y $\lambda_0^{H^T} > \lambda_0^{H^t}, \forall\ t \in \{T-1, \cdots, 1\}$, la próxima SL a descodificar sería H^{T^0}, y el actual conjunto de SLs viables sería $\{M^T, H^{T^1}, H^{T-1^0}, \cdots, H^{1^0}, L^{T^1}\}$.

3. Observe que la siguiente SL a L^{T^0} podría haber sido L^{T^1}. Siendo entonces el conjunto de SLs factibles $\{M^T, H^{T^0}, H^{T-1^0}, \cdots, H^{1^0}, L^{T^2}\}$

Esta idea fue implementada en el algoritmo OSLA (véase el Alg. 2.1). El cual se ejecuta a nivel de GOP, teniendo como parámetros de entrada la secuencia de $Q(T+$

1) SLs. El conjunto de SLs factibles para su adición al code-stream truncado en la progresión de SLs actual se encuentra en S. Para seleccionar la SL con mayor beneficio para la reconstrucción, S es ordenada en orden descendente por sus pendientes R/D (véase la Eq. 2.5). La mejor SL en cada iteración del algoritmo es almacenada en Λ, una lista que finalmente contiene SLs ordenadas por pendiente. Λ puede almacenarse en un segmento COM del encabezado del coef de L^T (en la SL L^{T^0} que es siempre la primera). A continuación, los clientes JPIP recuperan las capas de calidad de cada coef del GOP en el orden especificado en Λ.

Algoritmo 2.1 OSLA.

1. `for each` GOP:
2. $\Lambda = []; i = 0$ `// Lista vacía de SLs.`
3. $\Lambda[i++] =$ `input` L^{T^0}
4. $S =$ `input` $\{L^{T^1}, M^T, H^{T^0}, \cdots, H^{1^0}\}$ `// Siguientes posibles SLs.`
5. `while` $S \neq \emptyset$:
6. $s = \arg\max_{s \in S} (\lambda^s \geq \lambda^{s'} \; \forall s' \in S)$ `// Calcula cual es la máx.` λ^s.
7. $\Lambda[i++] = s$ `// Añade` s `(la SL con máx.` λ`) a` Λ.
8. $S = S \setminus \{s\}$ `// Elimina` s `de` S.
9. `if` $s == M^i$: `// Según` s `pertenezca a` M`,`L `o` H.
10. $S = S \cup \{M^{i-1}\}$ `// Añade la siguiente` M `a` S.
11. `else if` $s == L^{T^i}$:
12. $S = S \cup \{L^{T^{i-1}}\}$ `// Añade la siguiente SL de` L `a` S.
13. `else` $/* \; s == H^{t^i} \; */$:
14. $S = S \cup \{H^{t^{i-1}}\}$ `// Añade la siguiente SL de` H `a` S.
15. `output` Λ `// Salida: Lista ordenada de SLs.`

En la Fig. 2.5 se muestra un ejemplo del orden de las SLs en orden descendente de λ, suponiendo únicamente que L^{2^0} es la primera SL, suposición que está basada en la propia estructura creada por MCTF al generar las subbandas. Dado que las reconstrucciones de todas las imágenes están basadas en los datos de la subbanda L, no tiene sentido enviar información de otras subbandas sin haber transmitido previamente parte de L. Además la reconstrucción de L no se basa en la predicción del movimiento y tampoco tiene sentido enviar MVs antes que L. Este supuesto siempre se ha cumplido

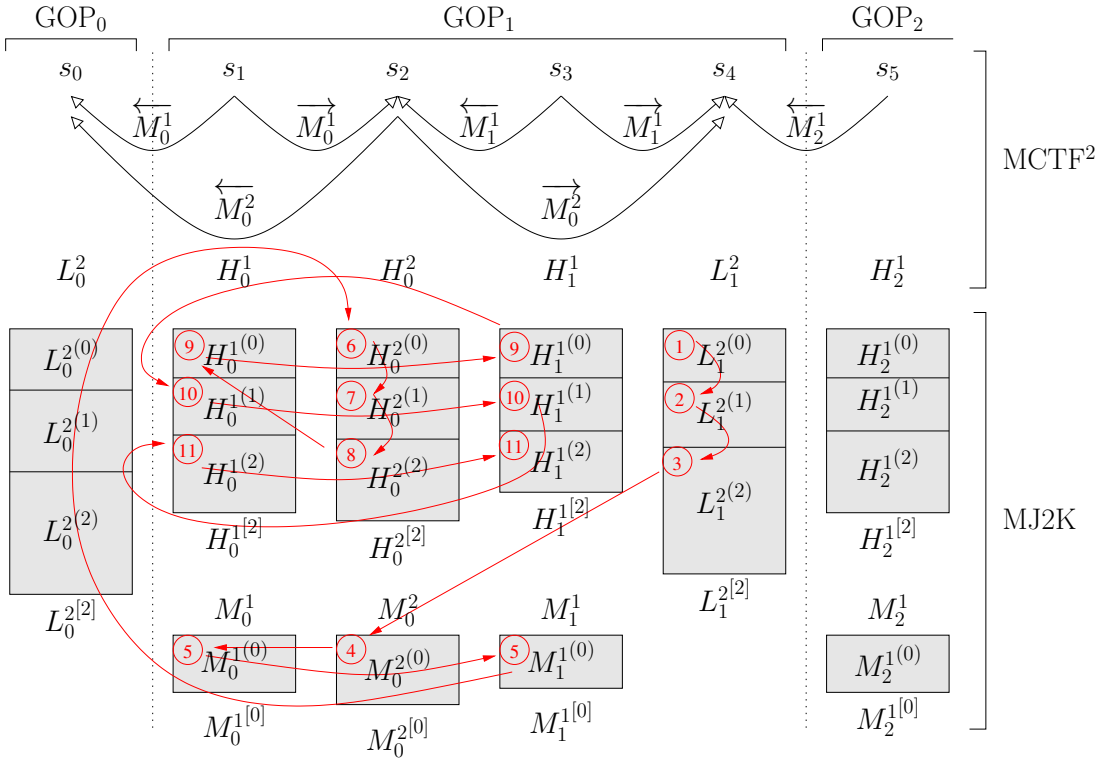


Figura 2.5: Ejemplo de esquema de ordenación de OSLA ($T = 2$ y $Q = 3$).

experimentalmente en todo experimento llevado a cabo. De forma que, en un escenario en el que el ancho de banda sólo permite la transmisión de una capa de calidad, la primera capa de la subbanda L, es la que proporciona un mayor rendimiento R/D en la reconstrucción.

El ejemplo mostrado en la Fig. 2.5 representa la progresión de SLs suponiendo que los cálculos de los RD-slopes resulten como sigue: $\lambda_0^{L^2} \geq \lambda_1^{L^2} \geq \lambda_2^{L^2} \geq \lambda^{M^2} \geq \lambda^{M^1} \geq \lambda_0^{H^2} \geq \lambda_1^{H^2} \geq \lambda_2^{H^2} \geq \lambda_0^{H^1} \geq \lambda_1^{H^1} \geq \lambda_1^{H^1}$.

2.2.2 Estimated-slope SL Allocation (ESLA)

El algoritmo ESLA usa las atenuaciones (véase la Sec. 2.1.2.1) para escalar los RD-slopes de cada SL del GOP, cuando estas pendientes se han determinado teniendo en cuenta únicamente la reconstrucción del correspondiente coef, no la reconstrucción del GOP completo como hace OSLA (observe que por esta razón, OSLA no necesita utilizar tales atenuaciones). Así, por ejemplo, un RD-slope para una SL de un coef de

la subbanda H^3 resultante de un MCTF[5] se divide entre $4{,}061$ respecto del RD-slope de L. En los casos en que hay más de un coef en una subbanda temporal, como en el ejemplo comentado, el promedio de todas las pendientes escaladas se utiliza para estimar la contribución de la SL correspondiente.

Esta idea fue implementada en ESLA (véase el Alg. 2.2), el cual devuelve la lista ordenada de SLs en Λ. Su funcionamiento es similar a OSLA. Para cada GOP la secuencia de entrada de $Q(T+1)$ SLs de S se ordenan en orden descendente según el valor de su RD-slope. La diferencia es que en OSLA las pendientes (λ) de las SLs son calculadas, mientras que ESLA las estima como un promedio ponderado de los RD-slopes de las SLs de los correspondientes coefs.

Si estos RD-slopes están predefinidos (en la compresión de los coefs se utiliza el mismo conjunto de slopes para todos ellos), ESLA puede ejecutarse en el lado del cliente sin necesidad de ninguna otra información sobre el R/D de las reconstrucciones. Esto significa que el cliente JPIP puede determinar el orden de las SLs para todos los GOPs de la secuencia después de recibir sólo los parámetros T, Q y conocer las atenuaciones de las subbandas (véase la Tab. 2.1). Observe que ninguno de estos parámetros depende de la secuencia de vídeo codificada. Por esta razón y porque ESLA tiene una complejidad computacional insignificante, ESLA es más adecuado que OSLA en escenarios de streaming en tiempo real.

Algoritmo 2.2 ESLA.

1. `for each` GOP:
2. $\quad\Lambda = []; i = 0$ `// Lista vacía de SLs.`
3. $\quad$`for each` $q \in \{0, \cdots, Q-1\}$:
4. $\quad\quad\Lambda[i++] = \texttt{input}\{\lambda_q^{H^T}, \cdots, \lambda_q^{H^1}\}$ `// Inicializa` Λ `con los` λ^H.
5. $\quad$`for each` $\lambda^k \in \Lambda$:
6. $\quad\quad\lambda^k = \lambda^k / \alpha^k$ `// Atenúa los` λ^H.
7. $\quad\Lambda[i++] = \texttt{input}\{\lambda_0^{L^T}, \cdots, \lambda_{Q-1}^{L^1}\}$ `// Añade a` Λ `los` λ^L.
8. $\quad$`sort_in_descending_order` Λ `// Ordena las SLs en` Λ `por` λ.
9. $\quad$`output` Λ `// Salida: Lista ordenada de SLs.`

La Fig. 2.6 representa un ejemplo de la selección de SLs que realiza ESLA. ESLA ordena las SLs en orden descendente de RD-slopes (referido a la calidad de reconstruc-

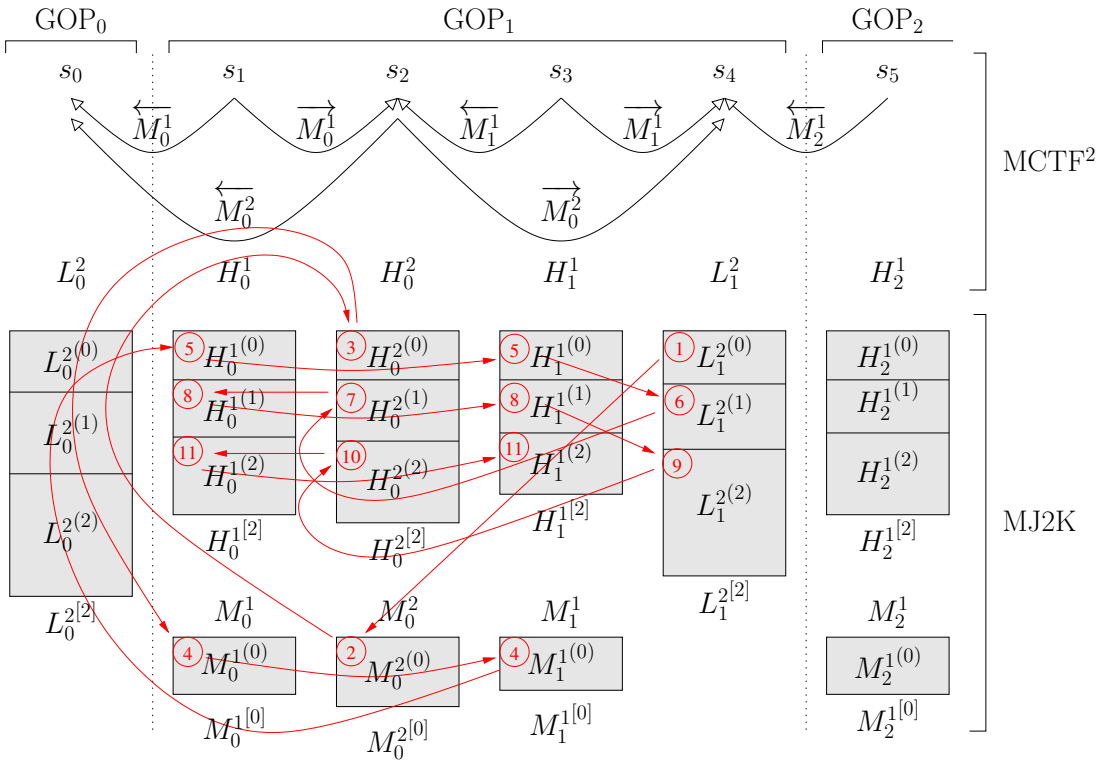


Figura 2.6: Ejemplo de esquema de ordenación de ESLA ($T = 2$ y $Q = 3$).

ción del coef), suponiendo que $\lambda^{M^T} \geq \lambda^{M^t}, \forall\, t \in \{T-1, \cdots, 1\}$, y por lo tanto los MVs serán transmitidos justo antes de la primera SL de su correspondiente subbanda H. En cuanto a las texturas, se supone que $\lambda_0^{L^2} \geq \lambda_0^{H^2} \geq \lambda_0^{H^1} \geq \lambda_1^{L^2} \geq \lambda_1^{H^2} \geq \lambda_1^{H^1} \geq \lambda_2^{L^2} \geq \lambda_2^{H^2} \geq \lambda_2^{H^1}$.

La teoría que sustenta nuestro trabajo es básica: debería existir, para cada vídeo codificado en MCJ2K, un orden óptimo para la transmisión de las SLs bastante cercano al determinado por nuestro algoritmo de rate-allocation ESLA, que se basa en que MCTF puede ser un proceso ortogonal o cercano a éste. Nótese, que el orden óptimo no viene determinado por ninguno de los algoritmos propuestos. Sin embargo, hemos encontrado que cuando MCTF presenta un comportamiento ortogonal (depende del vídeo), ESLA obtiene un rendimiento R/D cercano al que obtendríamos si minimizamos, capa por capa, el error de reconstrucción (esto es lo que hace OSLA). Observe que OSLA, basado en el cálculo de la mejor SL que pueda añadirse a un code-stream truncado, ha sido utilizado para evaluar el rendimiento de ESLA, nuestra propuesta de

rate-allocation durante la post-compresión para streaming en tiempo real.

2.3 Code-stream resultante de MCJ2K

La Fig. 2.7 muestra un ejemplo del code-stream resultante de MCJ2K, donde se detallan los GOPs, los coefs contenidos en éstos y las dependencias entre coefs por la descorrelación temporal hecha por la etapa MCTF. También se muestran las subbandas temporales divididas en sus SLs en la etapa de compresión. Y finalmente se representan las etapas de rate-allocation durante y después de la compresión.

Como puede apreciarse, un code-stream MCJ2K es una colección de texturas comprimidas (cada una compuesta por coefs) y subbandas de movimiento. Se han comprimido nueve imágenes, aunque sólo se muestran las seis primeras $\{V_0, \cdots, V_5\}$. El ejemplo mostrado se corresponde con una transformación MCTF de dos niveles ($T = 2$), indicado como MCTF^2, lo que implica un tamaño de GOP de cuatro imágenes ($G = 4$), excepto para el primero, el GOP$_0$ siempre tiene una sola imagen. Entonces en total, para las nueve imágenes se muestran 3 GOPs.

En la etapa MCTF de la Fig. 2.7, MCTF^2 transforma la secuencia de entrada V en 3 subbandas de texturas $\{L^2, H^2, H^1\}$ y 2 "subbandas" de movimiento $\{M^2, M^1\}$. L^2 es la subbanda de texturas de baja frecuencia y representa las componentes temporales de baja frecuencia de V. $\{H^2, H^1\}$ contienen las componentes temporales de alta frecuencia de V. $\{M^2, M^1\}$ almacenan una descripción del movimiento detectado en V. Las flechas en la etapa MCTF indican las dependencias de descodificación entre coefs. Cuando se aplica la transformación inversa, se genera una sucesión de niveles de resolución temporal cada vez mayores $\{L^2, L^1, L^0\}$. Por definición, $L^0 = V$.

En la etapa MJ2K+RA de la Fig. 2.7, MJ2K comprime los coefs usando una técnica de rate-allocation descrita en la Sec. 2.1.2.1, que selecciona el nivel de cuantificación para cada subbanda. Cada coef se indica como s_i^t, siendo i el índice del correspondiente coef dentro de su subbanda s, y t el nivel de dicha subbanda temporal. Cada SL de cada coef se representa como ${s_i}^{t^q}$, donde q indica el índice de la capa de calidad. Y en este contexto ${s_i}^{t^{(q)}}$ representa la calidad proporcionada por dicha SL y ${s_i}^{t^{[q]}}$ la calidad acumulada de todas las SLs hasta la capa q. En el ejemplo de la Fig. 2.7

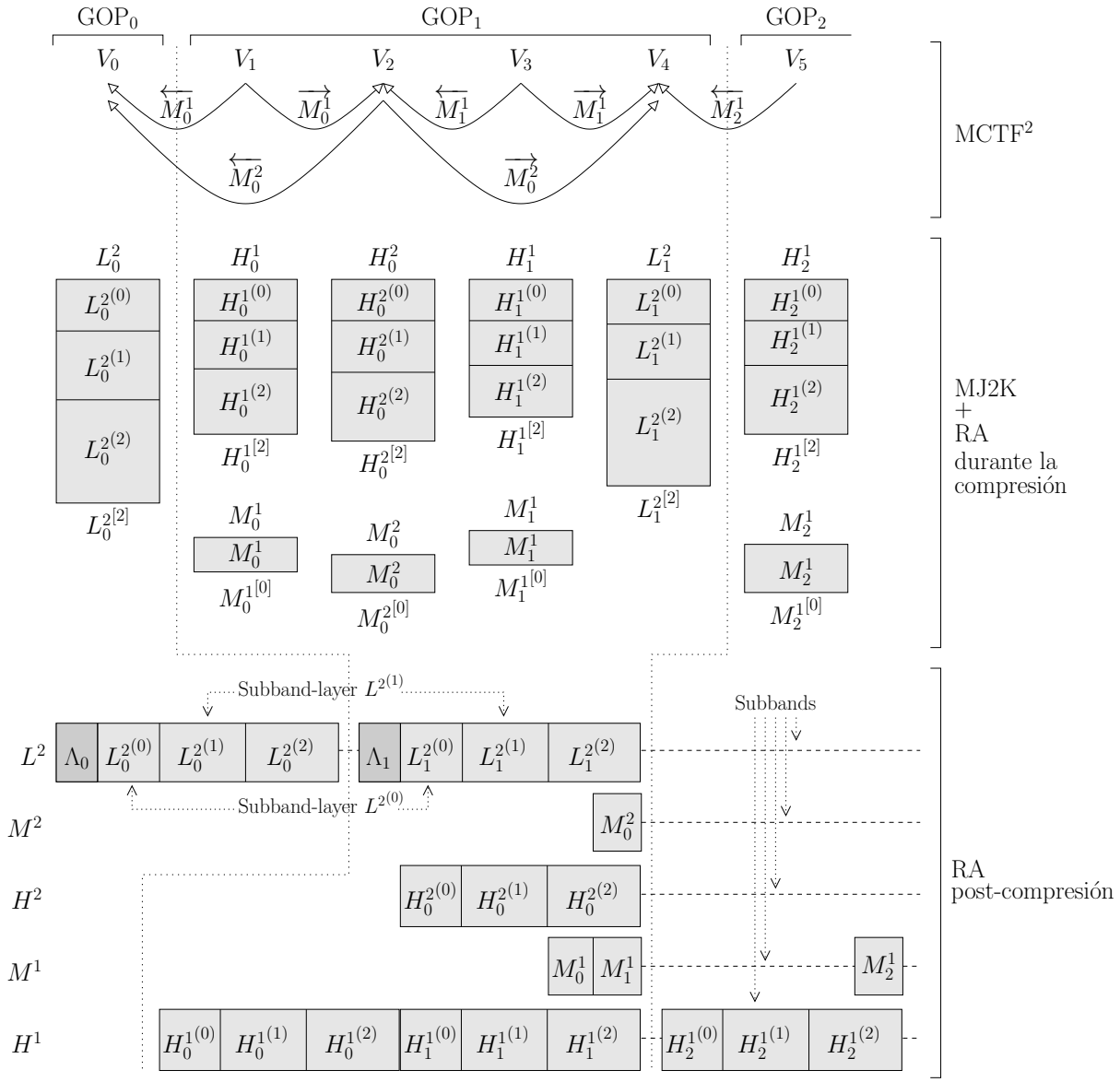


Figura 2.7: Ejemplo de code-stream MCJ2K para MCTF^2.

se han usado $Q = 3$ capas de calidad que se numeran según $\{0, \cdots, Q-1\}$, desde la capa base hasta la de mayor calidad, respectivamente.

Finalmente en la etapa RA (después de la compresión) de la Fig. 2.7, se aplican técnicas de rate-allocation que crean una lista ordenada Λ_g que almacena un orden que aumenta el rendimiento R/D de la reconstrucción parcial del code-stream del GOP g. Para $g = 0$ esta ordenación es trivial (adición consecutiva de las capas de calidad), dado que sólo tiene una imagen. Para el resto de GOPs se aplican las técnicas descritas en la Sec. 2.2.

2.4 Escalabilidad en MCJ2K

MCJ2K comprime una secuencia de imágenes y genera un code-stream que puede ser truncado a diferentes longitudes, proporcionando reconstrucciones escalables. Este repertorio de subbandas, campos de movimiento y capas de calidad hace posible un envío selectivo y óptimo del code-stream cuando es necesario truncarlo, proveyendo de escalabilidad en calidad, espacial y temporal. Así, en escenarios de streaming interactivo en tiempo real con ancho de banda limitado, sólo se transmitirá un subconjunto de SLs. Además, alojando el code-stream en servidores JPIP estándar, los clientes pueden usar WOIs.

De esta forma, en situaciones de transmisión real donde se produce un truncado del code-stream (incluso la falta completa de una subbanda), el descompresor necesita ser capaz de reconstruir el vídeo utilizando un code-stream incompleto. En nuestra propuesta, en caso de que falte uno o varios campos de movimiento, se utiliza la compensación de movimiento para interpolar los datos que faltan. Así, el descompresor interpola linealmente las imágenes correspondientes a los coefs que no se reciben, manteniendo constante la tasa de imágenes. Si los datos que faltan son una o varias subbandas de texturas, las texturas no recibidas se interpretan como gris, un valor de crominancia y luminancia situado entre los valores extremos de blanco o negro, y se mezclan con las texturas de los coefs que estén disponibles.

Las tres escalabilidades aquí presentadas son las más utilizadas. Para su exposición se usarán las Figs. 2.8 y 2.9. Debe tenerse en cuenta que cada subbanda temporal $\{L^T, H^T, H^{T-1}, \cdots, H^1\}$ es un code-stream diferente y aunque se conoce el número

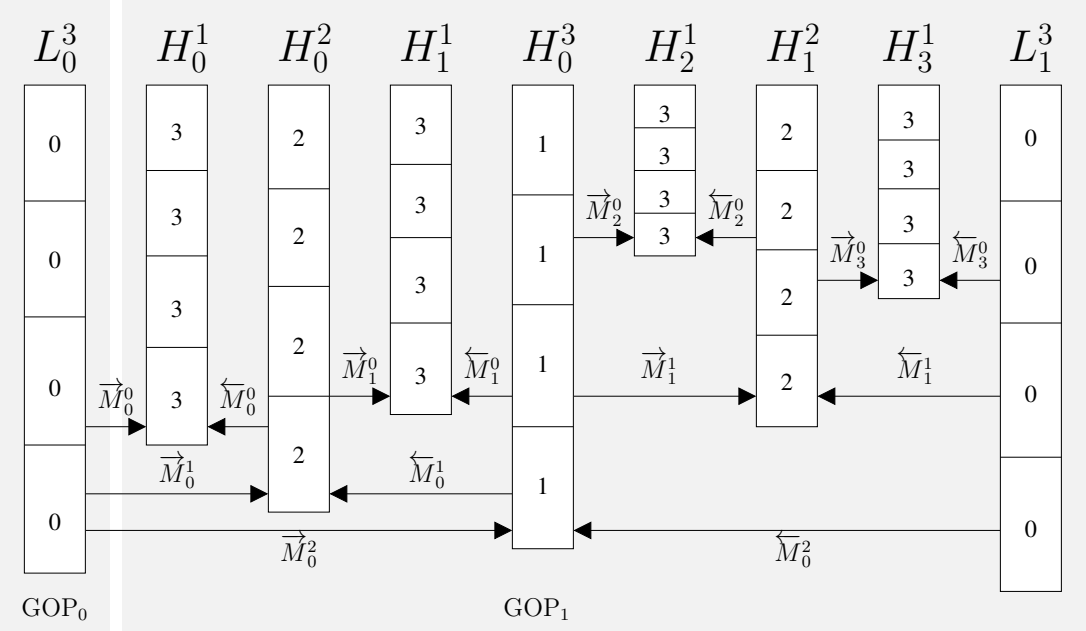


Figura 2.8: Escalabilidad temporal en MCJ2K.

de coefs en cada subbanda, cada imagen puede tener un rate distinto. Este rate está representado en estas figuras como un tamaño variable (a modo de ejemplo) de cada SL, que depende de la complejidad de sus texturas y del grado de éxito de la predicción de movimiento.

La distribución de datos en un code-stream MCJ2K permite explotar las tres escalabilidades (espacial, temporal y en calidad). Para ello, se usan tres de los 5 tipos de organización de los paquetes que componen las imágenes (LRCP, RLCP, RPCL, PCRL y CPRL), descritos en la Sec. 1.2.1.

La escalabilidad temporal

Se refiere a la posibilidad de descodificar un subconjunto de imágenes, generalmente siguiendo un esquema diádico. Interviene en el número de imágenes por segundo que se transmiten, es decir, el número de paquetes enviados por resolución y calidad. Las subbandas temporales deben ser descodificadas en orden para visualizar la secuencia de imágenes de forma cronológica.

En la Fig. 2.8 se muestran las subbandas temporales $\{L^3, H^3, H^2, H^1\}$, divididas en cada uno de los coefs que las componen y éstas en sus SLs correspondientes. Aquí las SLs están numeradas indicando el orden de descodificación para obtener la escalabilidad temporal. La mínima cantidad de imágenes que pueden ser descodificadas son dos, las correspondientes a $\{L_0^3, L_1^3\}$. El siguiente punto de truncado consiste en la adición de las SLs de $\{H_0^3\}$ para obtener 3 imágenes. El siguiente consiste en la adición de $\{H_0^2, H_1^2\}$ para obtener 5 imágenes en total. Para obtener el máximo frame/rate (en español, el número máximo de imágenes/segundo) se descodifica el GOP completo.

El acceso aleatorio a imágenes consiste en la descodificación de una sola imagen. Idealmente usando únicamente datos de dicha imagen (esto no está disponible para todas las imágenes en los codificadores con compensación de movimiento). En la Fig. 2.8 las flechas que enlazan unos coefs con otros indican la dependencia de la predicción del movimiento. Por tanto, para descodificar coefs con el uso mínimo de rate (usando sólo datos de las imágenes que finalmente obtendremos), el orden de descodificación debe ser el indicado en la numeración de las SLs en la figura, para la escalabilidad temporal.

En la Sec. 3.3.2 analizamos la sobrecarga del code-stream escalable de MCJ2K y en la Sec. 3.5.2 se evalúa el comportamiento en escenarios de acceso aleatorio a imagen en MCJ2K.

La escalabilidad en calidad

También llamada en SNR (Signal to Noise Ratio, en español ratio señal/ruido), hace posible el renderizado de un vídeo con una calidad proporcional a la cantidad de SLs descodificadas. Esta proporción no tiene por qué ser lineal, de hecho habitualmente no lo es.

En la Fig. 2.9 las subbandas temporales deben ser descodificadas usando el orden indicado en las SLs, es decir en orden secuencial en cada coef. Así el mínimo punto de truncado para la escalabilidad en calidad se corresponde con la descodificación de todas las primeras SLs de cada coef. El segundo punto de truncado se obtiene al adicionar la descodificación de todas las segundas SLs, y así sucesivamente.

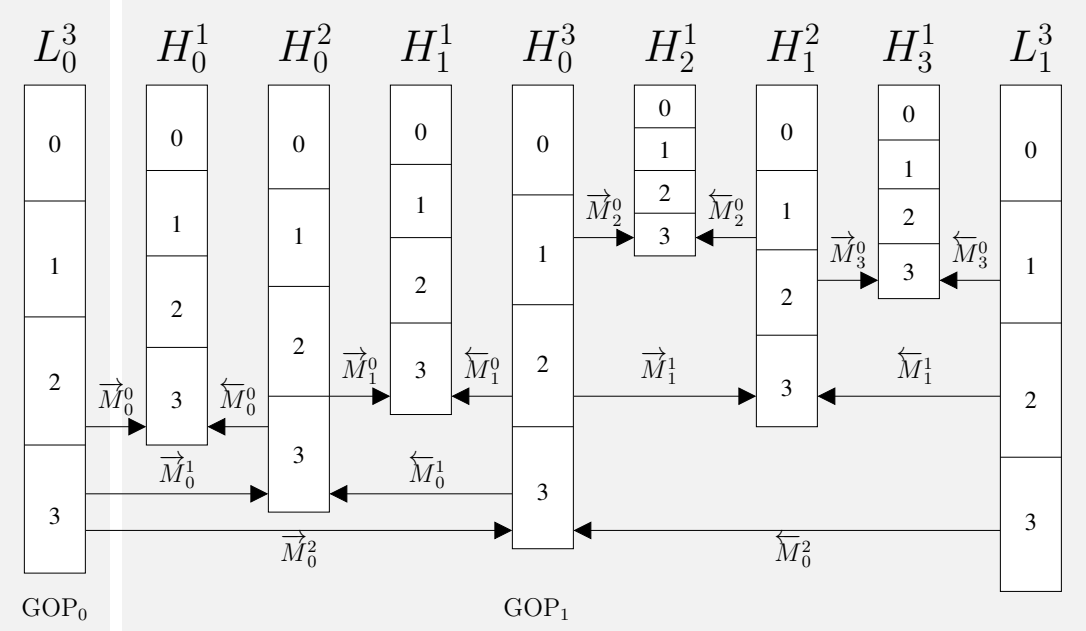


Figura 2.9: Escalabilidad en calidad y espacial en MCJ2K.

Para un grano más fino de la escalabilidad en calidad se debe seguir el orden de descodificación LRCP, donde todos los paquetes de cada SL se descodifican para cada resolution, component y precint, antes de comenzar la descodificación de la siguiente SL. La calidad mejora en todo el tile cada vez que se adiciona a la descodificación una nueva SL.

En la Sec. 3.3.1 se muestra la sobrecarga que supone cada capa de calidad, por un lado, el mayor rate derivado de ellas, y por otro, el mayor coste computacional en las técnicas de rate-allocation.

La escalabilidad espacial

Se utiliza para descodificar una versión a menor resolución de las imágenes. Se muestra la misma figura para la *escalabilidad espacial y en calidad* porque ambas necesitan organizaciones en los datos similares. Esto significa que, si se solicita una reconstrucción por niveles de resolución espacial, se debe usar el orden que genera la escalabilidad en calidad (LRCP) y descartar aquellos datos que pertenecen a subbandas DWT, que

generan una imagen de mayor resolución que la actualmente reconstruida. El orden utilizado es RLCP. Este tipo de ordenación consiste en que el primer paquete contiene una capa de calidad de un precint de un coef, a baja resolución, el segundo y los siguientes contienen información que amplía la resolución hasta alcanzar el máximo codificado.

Capítulo

3

Evaluación

Este capítulo muestra un estudio de la eficiencia de MCJ2K describiendo los resultados experimentales de nuestra propuesta. En la Sec. 3.1 se presenta el conjunto de vídeos y códecs (junto a las configuraciones usadas en ellos) para la evaluación de MCJ2K. En la Sec. 3.2 se muestra una breve reseña del impacto que la estimación y compensación de movimiento tiene en la codificación de vídeo en MCJ2K comparado a MJ2K. En la Sec. 3.3 se muestra la diferencia en términos de R/D entre la transmisión progresiva y no-progresiva de MCJ2K. En las Sec. 3.4 y Sec. 3.5 se han evaluado las soluciones de rate-allocation de MCJ2K, comparando su rendimiento R/D con otros códecs estándar.

3.1 Métodos y materiales

En la evaluación experimental se utilizaron varios vídeos, que en conjunto forman un banco de pruebas variado en resoluciones y complejidad en la predictibilidad del movimiento. En el resto del capítulo se muestran los resultados más representativos, evitando gráficas redundantes. Las secuencias son:

1. *Mobile:* Un vídeo de baja resolución con movimiento complejo (movimiento difícil de predecir).
   YUV 4:2:0, 352×288 píxeles, 30 Hz.

2. *Container:* Un vídeo de baja resolución con movimiento simple o predecible.
   YUV 4:2:0, 352×288 píxeles, 30 Hz.

3. *Crew:* Un vídeo de resolución media con movimiento complejo.
   YUV 4:2:0, 704×576 píxeles, 60 Hz.

4. *City:* Un vídeo de resolución media con movimiento simple.
   YUV 4:2:0, 704×576 píxeles, 30 Hz.

5. *Parkrun:* Un vídeo de alta resolución con movimiento complejo.
   YUV 4:2:0, 1280×720 píxeles, 50 Hz.

6. *CrowdRun:* Un vídeo de alta resolución con un alto grado de movimiento.
   YUV 4:2:0, 1920×1080 píxeles, 50 Hz.

7. *ReadySetGo:* Un vídeo de muy alta resolución con alto grado de movimiento.
   YUV 4:2:0, 3840×2160 píxeles, 120 Hz.

8. *Sun:* Una secuencia de imágenes del Sol de resolución ultra-alta con sólo una pequeña cantidad de movimiento (que sin embargo, es complejo de predecir debido al movimiento aleatorio de las llamaradas solares). Es monocromático debido a que representa sólo una frecuencia del espectro irradiado por el Sol.
   YUV 4:0:0, 4096×4096 píxeles, 1/30 Hz.

Los códecs de vídeo usados para los experimentos, junto a sus configuraciones son:

1. *MCJ2K:* Códec de vídeo escalable propuesto.

2. *AVC:* Códec de vídeo no escalable.
   Configuración: *–profile high–preset placebo–tune psnr*

3. *HEVC:* Códec de vídeo no escalable, sucesor de H.264-AVC.
   Configuración: *trunk/cfg/encode_randomaccess_main.cfg*

4. *SHVC:* Códec de vídeo escalable.
   Configuración 1: *branches/SHM-dev/cfg/encoder_randomaccess_scalable.cfg*
   Configuración 2: *branches/SHM-dev/cfg/misc/layers8.cfg*

5. *MPEG-2:* Códec de vídeo no escalable.

En todos los experimentos se comprimieron 129 imágenes. Respecto a la predicción del movimiento el área de búsqueda fue de 4 píxeles sin usar precisión sub-píxel. El tamaño de macro-bloque fue de 32×32 para *Mobile*, *Container*, *Crew* y *City*;

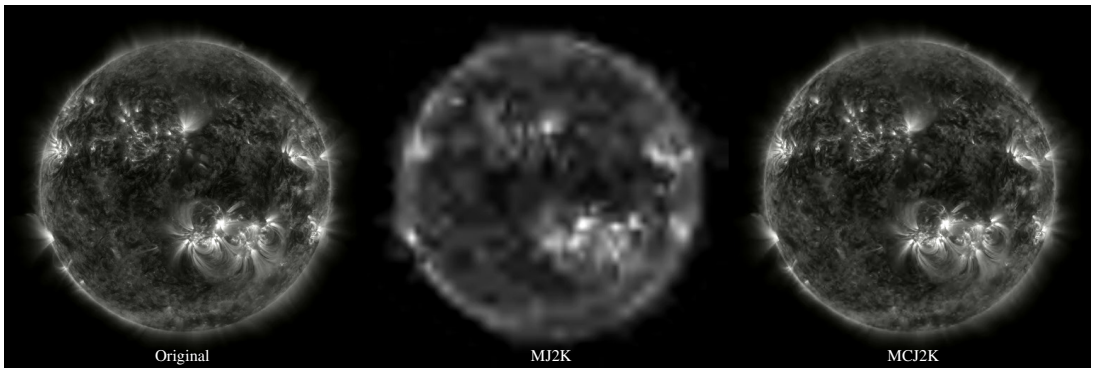


Figura 3.1: Reconstrucción de una imagen de *Sun*, usando MJ2K y MCJ2K, a bit-rates similares.

64×64 para *Parkrun*, *CrowdRun* y *ReadySetGo*; y finalmente 128×128 para *Sun*. Los parámetros utilizados para la compresión han sido de: 5 niveles de DWT, sin partición de precint y code-blocks de 64×64 coeficientes. El número de capas de calidad para los coefs fue de $Q = 8$, lo que proporciona un buen equilibrio entre el rendimiento de la compresión y la granularidad (útil en las técnicas de rate-allocation). En el caso de los datos de movimiento, sólo se usó una capa de calidad ($Q = 1$).

3.2 Impacto de la ME/MC en MCJ2K

La Fig. 3.1 muestra la reconstrucción de una imagen de la secuencia *Sun* descomprimida con MJ2K y MCJ2K, a similares bit-rate. Izquierda: imagen original con 512×512 píxeles y una cadencia de $\frac{1}{12}$ imágenes/segundo. Centro: misma imagen (progresivamente) descomprimida con MJ2K a $0,08$ Kbps. Derecha: misma imagen (progresivamente) descomprimida con MCJ2K a $0,04$ Kbps. Crédito de la imagen: NASA/SDO/AIA. Como puede observarse MCJ2K consigue menor distorsión con la mitad de uso del ancho de banda.

3.3 Transmisión progresiva vs. no-prog. en MCJ2K

En la Fig. 3.2 se muestra una comparativa entre una transmisión progresiva y no-progresiva, mostrando un amplio rango de bit-rate que expone el resultado R/D de las reconstrucciones bajo restricciones de ancho de banda. El rate está expresado en Kbps y la distorsión en PSNR (Peak Signal-to-Noise Ratio, en español pico del ratio ruido/señal, medido en dB). Ambas curvas se obtienen de la codificación de vídeo utilizando MCJ2K para 5 niveles de resolución temporal.

La curva progresiva llamada OSLA-MCJ2K en la Fig. 3.2, representa el mismo code-stream con distinto truncado. Cada punto en esta curva, indica la adición de una SL al code-stream truncado. El último punto de la curva (el de mayor rate) muestra el R/D del code-stream completo (8 SLs por subbanda, esto es $Q = 8$) descodificándose todas sus SLs y MVs. Para la elección del orden de progresión de las SLs se ha usado la técnica de rate-allocation OSLA (véase la Sec. 2.2.1).

En la curva no-progresiva de la Fig. 3.2 cada punto de la curva R/D representa un code-stream completo, que supone otra codificación del vídeo con un valor de cuantificación (q) distinto. El primer punto de la curva R/D (el de menor rate), representa una descodificación de un code-stream completo que utiliza un valor de cuantificación alto, que disminuye en cada nueva codificación hasta llegar al menor valor de q usado (punto con mayor rate).

Observe que a baja tasa de bits, en ningún caso la transmisión no-progresiva puede lograr un mejor rendimiento R/D que la progresiva. El streaming no-progresivo no puede aportar al cliente un code-stream con un bit-rate tan pequeño como en el streaming progresivo, incluso si aumentamos la cuantificación al máximo. En la Fig. 3.2 pueden observarse los siguientes ejemplos, del mínimo bit-rate necesario en el streaming no progresivo: 100 Kbps (Kilobits/segundo) en *City*, 250 en *Parkrun*, 900 en *Crowdrun* y $0,25$ en *Sun*. En la Tab. 3.1 se muestran los Kbps necesarios para el streaming de vídeo comparando ambos escenarios. El streaming progresivo de MCJ2K es capaz de transmitir vídeo con un ancho de banda casi 50 veces menor en algunos casos.

De estos resultados podemos concluir que la transmisión progresiva es especialmente útil a una tasa de bits baja, debido a que la transmisión no-progresiva (1) tiene una baja capacidad de compresión y (2) la distorsión es mayor que en la versión progresiva. En ambos casos ocurre que al intentar comprimir al máximo la versión

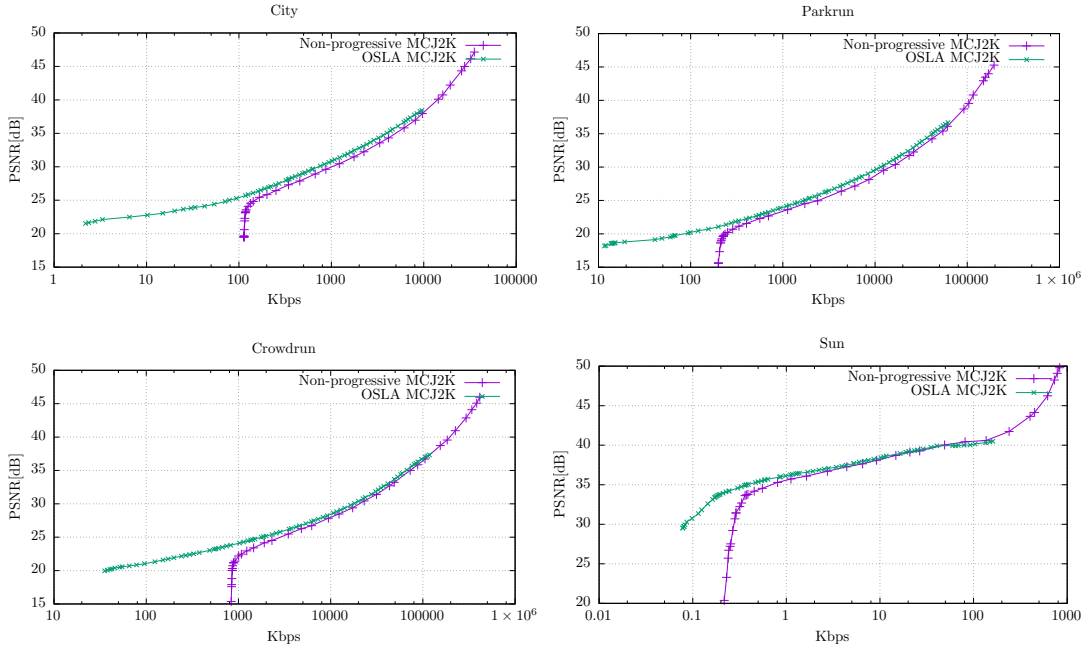


Figura 3.2: Transmisión progresiva vs. no-progresiva en MCJ2K.

	City	*Parkrun*	*Crowdrun*	*Sun*	
No-progresiva	120	250	900	0,25	Kbps
Progressiva	2,5	12	38	0,08	Kbps
	x48	x21	x23	x3	Diferencia

Tabla 3.1: Transmisión progresiva vs. no-progresiva a bajo bit-rate.

no-progresiva para alcanzar el bit-rate de la progresiva, se incrementa la cantidad de cuantificación haciéndola tan alta que elimina la información útil de las texturas. Esto provoca que sólo queden las cabeceras de los paquetes JPEG2000 que no aportan mayor calidad en la reconstrucción. Dado que a bajo bit-rate la versión no-progresiva transmite un mayor número de cabeceras (las de todo el code-stream) y MVs sin texturas que mover (capas vacías), la versión no-progresiva tiene una gran cantidad de información no útil.

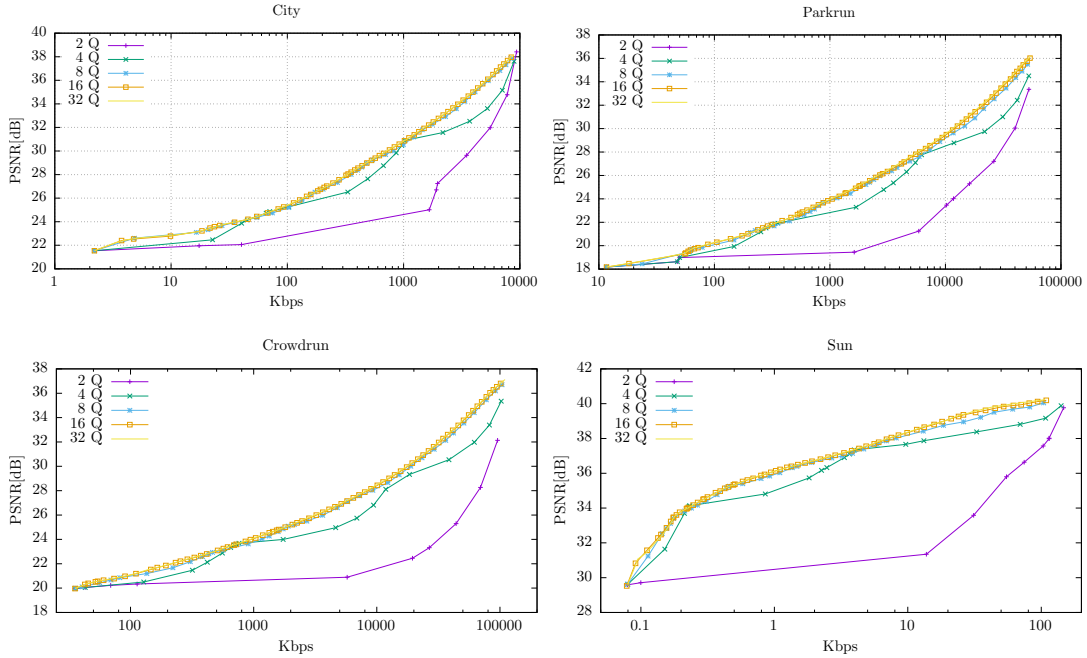


Figura 3.3: Puntos de truncado y sobrecarga variando Q, con q sin atenuar.

3.3.1 Coste de la escalabilidad en calidad

El número de capas de calidad tiene una gran influencia en el rendimiento de la transmisión de un code-stream MCJ2K porque cuanto mayor sea la granularidad, mayor rate-control puede llevar a cabo el cliente JPIP. En la Fig. 3.3 se muestra la codificación de varios vídeos usando distinto número de capas por subbanda (Q), siendo éstos 2, 4, 8, 16 y 32. Los code-streams producidos son escalables y las curvas R/D mostradas se extraen de la descodificación progresiva del code-stream, capa a capa.

La Fig. 3.3 muestra mejores resultados cuando Q es mayor, aunque no hay una diferencia significativa entre usar 8, 16 y 32 capas de calidad, es decir, para valores mayores de Q el rendimiento incluso se degrada debido al aumento del número de paquetes J2K y por tanto, de cabeceras que crecen linealmente con el aumento de dichos paquetes.

Nótese que en la descodificación completa de los code-streams (punto de mayor rate en cada curva R/D) no se obtiene el mismo bit-rate para distintos valores de Q. Esto se debe a que cada code-stream no está comprimido con las mismas cuantificaciones

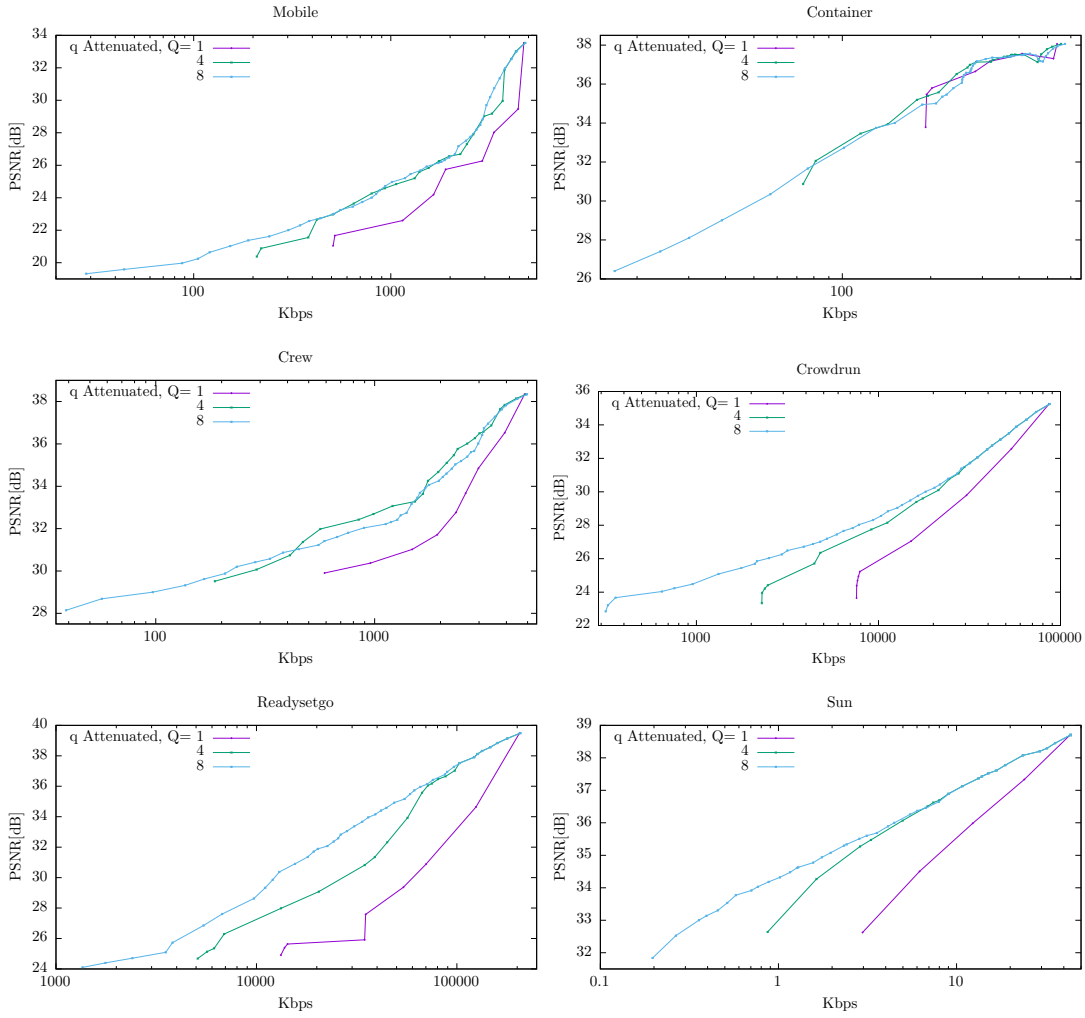


Figura 3.4: Puntos de truncado y sobrecarga variando Q, con q atenuadas.

q (mediante el correspondiente slope), ya que ni siquiera tienen el mismo número de ellas (número de capas), de hecho éste es el objetivo del experimento.

La experimentación sugiere que 8 capas de calidad constituyen un buen compromiso entre una notable granularidad del code-stream y el coste computacional del proceso de rate-allocation. Usando $Q = 8$ capas de calidad para cada subbanda, en el caso de que usemos 5 niveles de resolución temporal, suponen 40 puntos de truncado disponibles en el code-stream (más los 4 campos de movimiento asociados a cada una de las subbandas H). En total, el servidor tiene un total de 44 partes en las que el code-stream puede ser truncado durante la transmisión para adaptarse a los cambios de ancho de banda, por ejemplo. Por esta razón, $Q = 8$ será el número de capas de calidad utilizadas para el resto de los experimentos (a menos que se especifique lo contrario).

En la Fig. 3.3 las cuantificaciones utilizadas han sido seleccionadas tras especificar los siguientes parámetros: (1) el número de capas Q, (2) el número de TRLs y (3) al menos una cantidad de cuantificación q. Éste q establece la cuantificación de la última capa de calidad (la de mayor calidad, es decir, la menos cuantificada), y a partir de ella se establecen los valores de los q restantes de forma equidistante (véase la Sec. 2.1.2.1). Sin embargo, para poder comparar las curvas R/D en un rango de bit-rate similar entre ellas es necesario incrementar el slope-step a medida que la curva tenga menos capas, para así llegar a cuantificaciones máximas y mínimas similares en todas las curvas independientemente de su número de capas.

En la Fig. 3.4 los slope-steps se calculan en base a las atenuaciones presentadas en la Sec. 2.1.2.1, que establecen la cantidad de cuantificación para cada SL a partir de un q dado. Éste es el motivo por el que en esta figura los últimos puntos (alto bit-rate) de las curvas R/D coincidan, pero los primeros (bajo bit-rate) no.

3.3.2 Coste de la escalabilidad temporal

El estándar MJ2K puede proporcionar una escalabilidad temporal sin restricciones. Sin embargo, los ratios de compresión proporcionados por MJ2K son bajos comparados con otros códecs de vídeo que sí explotan la redundancia temporal. MCJ2K usa para esto MCTF y puede alcanzar ratios de compresión competitivos, dependiendo de la

predictibilidad del movimiento.

En la Fig. 3.5 se muestra el códec MCJ2K utilizando la compensación de movimiento multinivel, en un rango de número de TRLs desde 7 hasta 2. Un valor de 1 TRL implica que no se aplica ninguna descorrelación temporal (lo que se corresponde con MJ2K). El tamaño del GOP G indica el número de imágenes del GOP, es decir, las imágenes entre las cuales se realiza la estimación de movimiento.

La Fig. 3.5 muestra el rendimiento de MCJ2K comparado con el de MJ2K para diferentes tamaños de GOP. Cada vídeo se comprimió una vez y se descomprimió de forma progresiva, ordenando las capas de subbanda usando OSLA. MCJ2K fue en la mayoría de los casos superior a MJ2K, dependiendo de la correlación temporal encontrada en cada vídeo. Por ejemplo, MCTF es muy eficiente en *Container*, en el que se puede ver que, a 300 Kbps MCJ2K tiene 10 dB menos de distorsión que MJ2K, debido a que la estimación de movimiento resulta muy efectiva. Por el contrario, en el caso de *Sun*, donde el movimiento de las llamaradas solares representa un movimiento no predecible, el uso de MCTF es contraproducente. En el caso de *ReadySetGo*, en el que MCTF no es capaz de generar predicciones precisas, el uso de un tamaño de GOP superior a 4 no aumenta la calidad de las reconstrucciones.

La predictibilidad del movimiento en una secuencia cambia según G. En términos generales, a mayor G, menor probabilidad existirá de que el movimiento sea predecible entre imágenes de la secuencia que se encuentran más alejadas en el tiempo. Debido a esto, en secuencias con movimiento predecible, el rendimiento R/D de su codificación se beneficia al aumentar G dado que la predicción de movimiento actuará sobre un mayor número de imágenes, pudiendo explotar una mayor correlación temporal en el vídeo. En secuencias con movimiento no/poco predecible, el rendimiento R/D se beneficia al disminuir G, estableciendo menor distancia temporal entre las imágenes sobre las que se buscan similitudes. Por lo tanto, el tamaño del GOP tiene un alto impacto en el rendimiento de MCJ2K y es un parámetro que debe ser optimizado para cada secuencia de vídeo. Por los resultados observados, 5 TRLs es un compromiso satisfactorio.

En cualquier caso, debe tenerse en cuenta que el modelo de estimación de movimiento usado por MCJ2K es muy básico. Predictores de movimiento más avanzados,

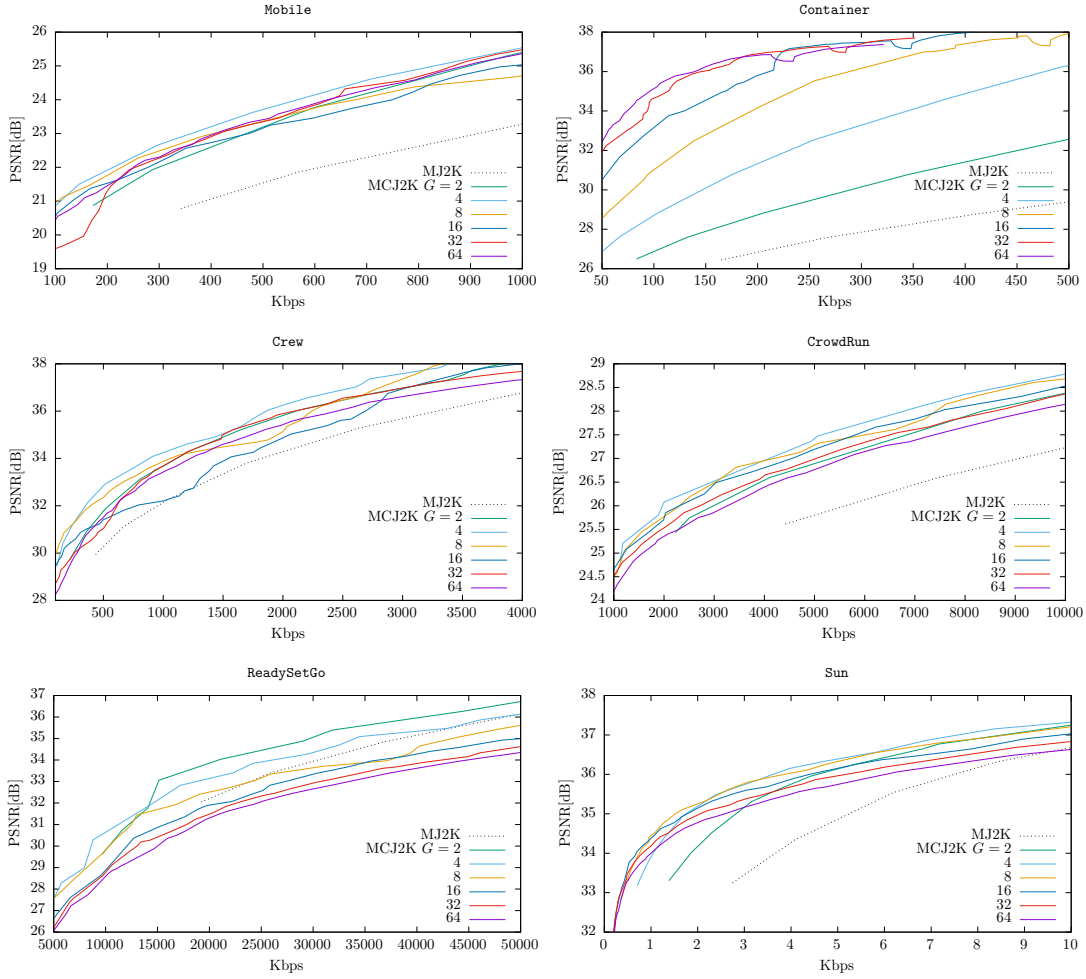


Figura 3.5: MCJ2K vs. MJ2K para diferentes valores de G y secuencias.

como los utilizados en los últimos estándares de codificación de vídeo (por ej. SHVC o HEVC), harían posible el uso de G mayores y, por lo tanto, mayores ratios de compresión.

3.4 Evaluación de RA durante la compresión

El rate control llevado a cabo por las atenuaciones se evalúa a continuación, mostrando casos de estudio del impacto, en términos de R/D, proporcionado por la utilización de las atenuaciones. En la Fig. 3.6 (arriba) se muestra la curva R/D para diferentes valores de G; (abajo) la mejora medida en PSNR (disminución de la distorsión), al utilizar las

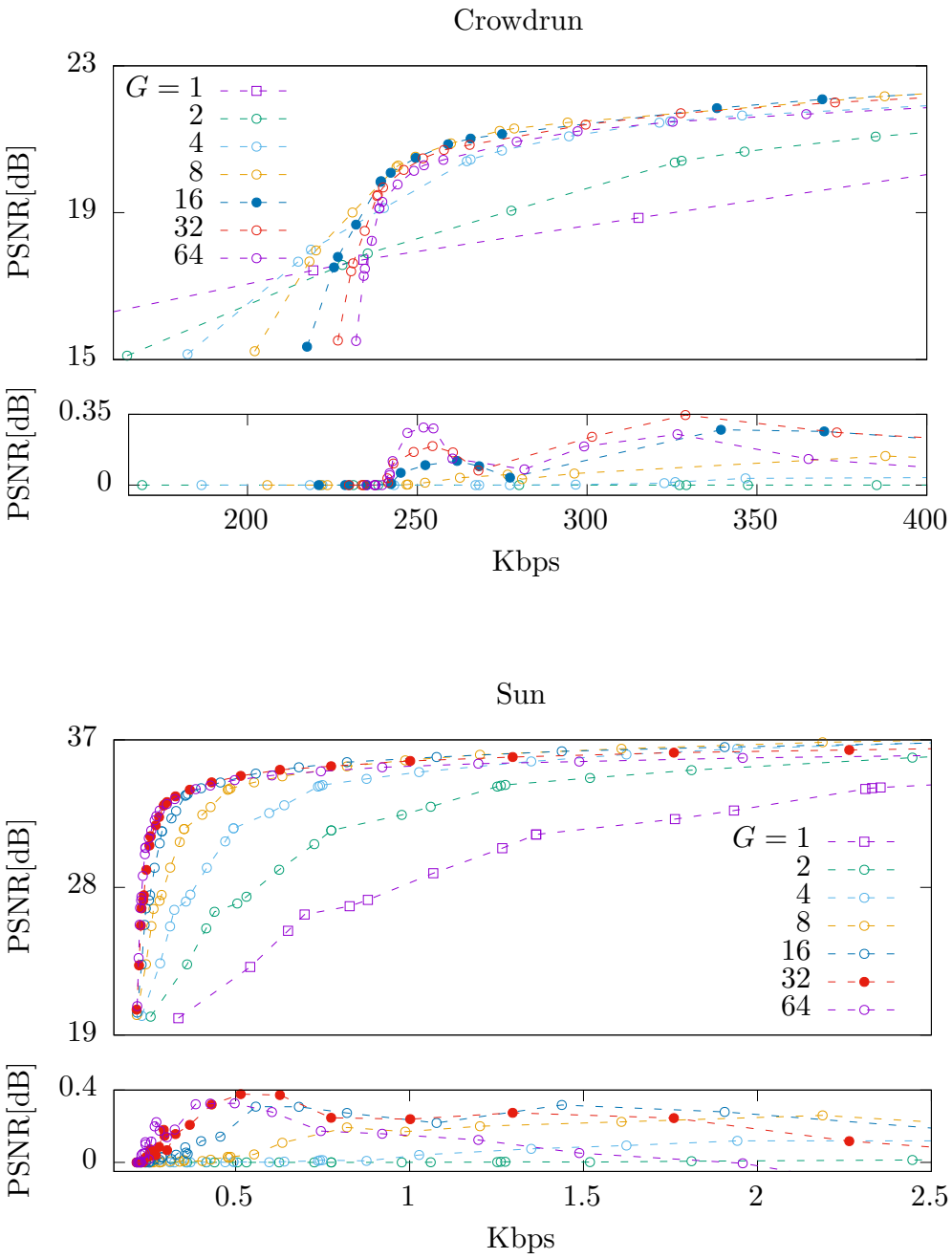


Figura 3.6: Impacto del uso de atenuaciones, para distintos valores de G.

atenuaciones. Observe que cada punto de truncado de la curva R/D (gráfica de arriba), se relaciona con su análogo verticalmente en la gráfica de abajo. El impacto del uso de las atenuaciones depende de las siguientes variables:

1. El número de niveles de resolución temporal, porque a cada SL le corresponde una cantidad diferente de atenuación dependiendo de la subbanda a la que pertenece, véase la Sec. 2.1.2.1.

2. La predictibilidad del movimiento en el vídeo o, visto de otra forma, del éxito en la estimación del movimiento, ya que influye directamente en el R/D de las reconstrucciones. Además según la predictibilidad del movimiento se puede seleccionar, en tiempo de compresión, el número de TRLs, parámetro relacionado a G y cuya relación con el éxito en la estimación de movimiento ha sido descrita en la Sec. 3.3.2.

3. La cantidad de cuantificación (slope), ya que un nivel de cuantificación demasiado elevado impide que las atenuaciones puedan aplicarse con éxito. Observe que al atenuar una subbanda, se aumenta su cuantificación, por tanto, si partimos de un nivel de cuantificación muy alto del que apenas resultan datos útiles, mayores atenuaciones en el resto de subbandas provocan cantidades información de cada una de ellas casi nulas, que no mantienen las proporciones que intentan establecer los índices de atenuación.

Para el resto de los experimentos de ese capítulo se utilizan q atenuadas, ya que producen una mejora (alrededor de $0,35$ dB) en términos de R/D, de forma estable en todos los casos de estudio evaluados. Además se trata de una técnica de rate-allocation sin coste computacional, que es completamente independiente de las arquitecturas de transmisión cliente/servidor.

3.5 Evaluación de RA en post-compresión

El rendimiento de las técnicas de rate-allocation propuestas en tiempo de post-compresión son evaluadas a continuación. Su comportamiento se diferencia después de que la progresión de SLs en el streaming comience por la primera capa de L, tras lo cuál entran en juego los distintos algoritmos de OSLA y ESLA.

Por un lado, para la segunda SL a transmitir, OSLA siempre elige la mejor posible, por lo que siempre inicia la transmisión mejor o igual que ESLA en términos de R/D. Por otro lado, en el resto del rango de bit-rate (para un ancho de banda que pueda albergar más de dos SLs), cada algoritmo puede tomar ventaja sobre el otro. Esto se debe a que OSLA tiene un rendimiento sub-óptimo, ya que sólo calcula la siguiente mejor SL (sólo una SL en cada iteración). El algoritmo de OSLA puede tener una selección óptima de SLs, simplemente iterando OSLA recursivamente para cada SL del code-stream (no sólo la siguiente SL posible) hasta llegar a un bit-rate dado. Nótese que un orden óptimo para un bit-rate no tiene por qué ser un subconjunto de SLs (en el mismo orden), de otro orden óptimo de un bit-rate mayor. Es decir, no tiene por qué existir un orden óptimo para cualquier bit-rate.

La cuestión es que la elección de una SL, entre el banco de posibles candidatas de SLs, influye en la elección de las posteriores, dado el proceso acumulativo de reconstrucción del vídeo, basado en adición de SLs. Así, puede ocurrir que al evaluar una única SL por subbanda, se obtenga un resultado diferente (elección de una SL), que si se evalúan varias SLs por subbanda antes de decidirse por una. Por lo tanto, el orden de las SLs resultante puede ser distinto en función del número de puntos de truncado en la progresión del streaming evaluados.

Sin embargo, la implementación presentada obtiene buenos resultados simplemente iterando para una única SL. La complejidad computacional de OSLA iterando para una SL por subbanda viene dada por la reconstrucción y el cálculo de la distorsión relativa de SLs $\times$ TRLs veces por vídeo, lo que ya lo hace inviable para las transmisiones en tiempo real. El objetivo de OSLA ha sido proporcionar una referencia para ayudar a evaluar el rendimiento de una solución de rate-allocation, ESLA, que puede ser ejecutada sin coste computacional, ni durante la codificación del vídeo ni durante su streaming.

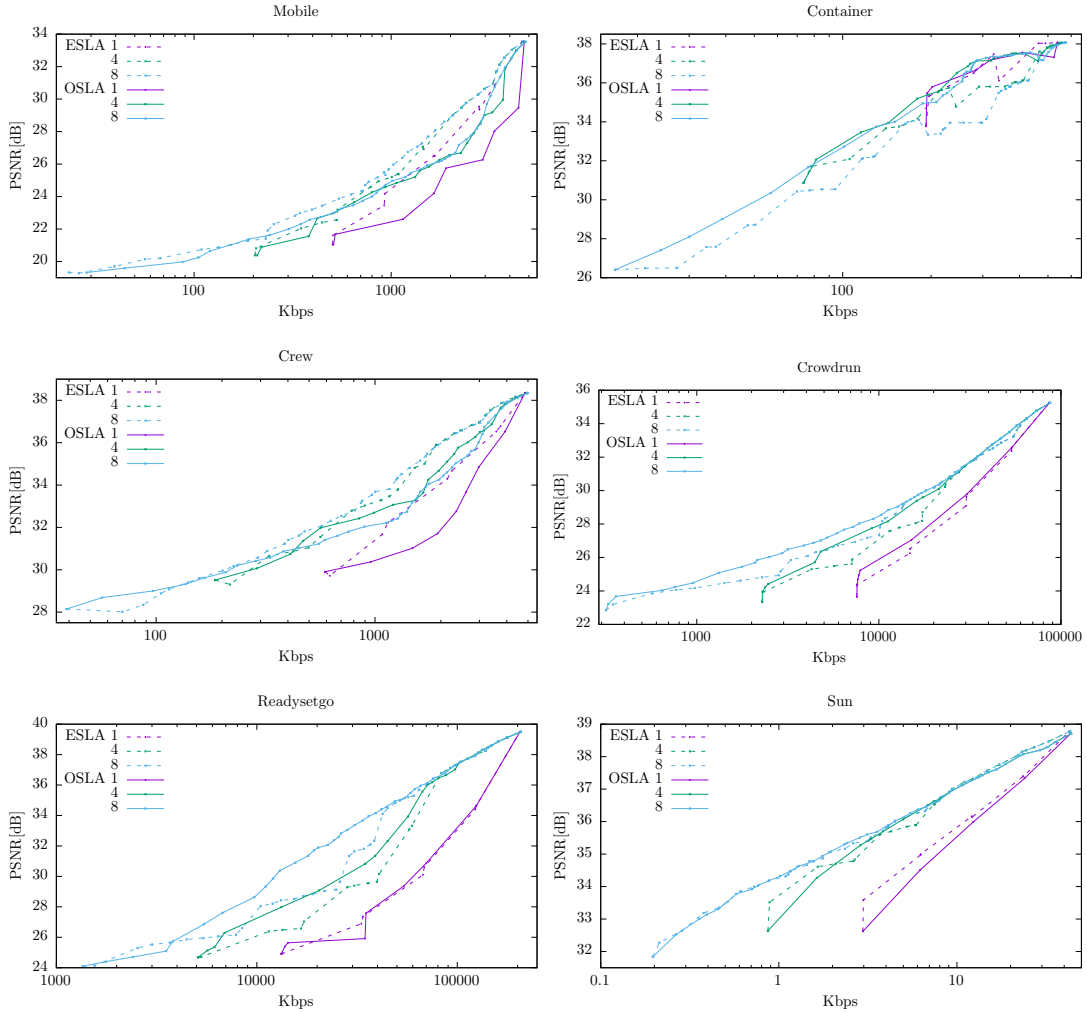


Figura 3.7: Puntos de truncado de OSLA y ESLA, para distintos valores de Q.

3.5.1 OSLA vs. ESLA

En la Fig. 3.7 la diferencia en el rendimiento de R/D entre OSLA y ESLA son menores según aumenta Q, siempre y cuando MCTF tenga un comportamiento ortogonal (sirva como ejemplo el vídeo Sun). A mayor Q, el algoritmo ESLA tiene mayor granularidad para mantenerse ajustado a las proporciones entre subbandas dadas por las atenuaciones. Es decir, al transmitir capa a capa, si una capa transmitida tiene un bit-rate muy alto, su envío provoca una descompensación entre las proporciones que proponen las atenuaciones entre subbandas, hasta que se vuelve a enviar un nueva capa.

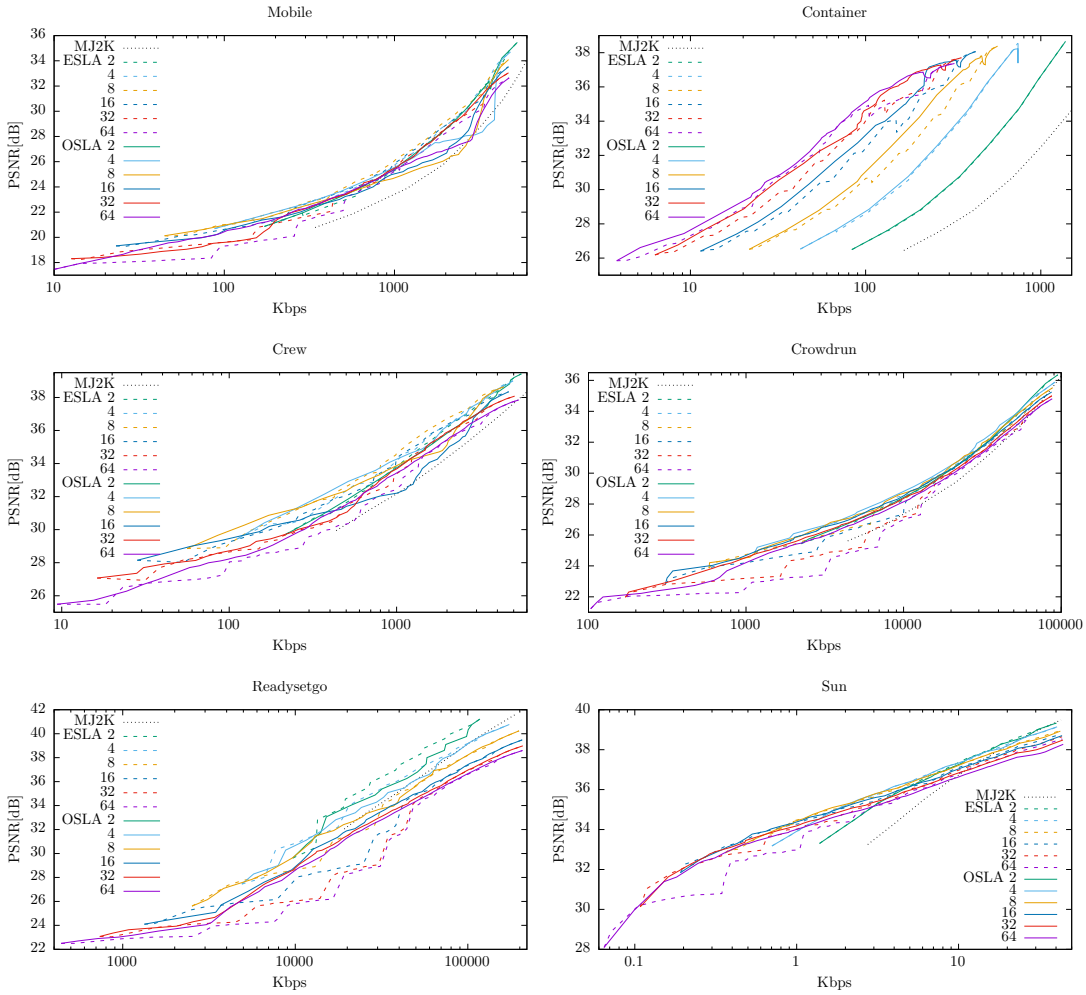


Figura 3.8: MCJ2K (usando OSLA vs. ESLA) vs. MJ2K, para diferentes valores de G y secuencias.

La Fig. 3.8 compara el rendimiento R/D entre el códec MJ2K y MCJ2K en varios niveles de resolución temporal, desde en el caso de MJ2K ($G = 1$), hasta 7 niveles de resolución temporal ($G = 64$). Esta figura permite comparar el diferente comportamiento de ESLA y OSLA según el número de TRLs. En los vídeos con movimiento complejo (*Crowdrun* y *Readysetgo*) o de alta predictibilidad (*Container*), es cuando MCTF tiene un grado de ortogonalidad menor y en consecuencia, donde destacan las diferencias entre ambas técnicas.

OSLA no acusa en su rendimiento R/D estas situaciones, debido a que su ordenación es calculada sobre cada caso específicamente. ESLA se basa únicamente en la distribución de energía realizada por MCTF suponiendo su ortogonalidad. Si la estimación de movimiento ha sido demasiado compleja, los residuos estarán sobredimensionados, para corregir el resultado pobre o incluso equívoco de los MVs.

Si la estimación de movimiento ha sido muy exitosa, MCTF deja de tener también un comportamiento ortogonal, ya que una pequeña cantidad de MVs (o texturas) puede disminuir significativamente la distorsión. En consecuencia las atenuaciones calculadas se alejan de estos casos de estudio.

En la Fig. 3.9 se muestra sólo el ejemplo más representativo de OSLA y ESLA, para que su visualización sea lo más clara posible. Utilizando la información proporcionada por los experimentos anteriores, se ha seleccionado un tamaño de GOP adecuado para cada secuencia y se compara el rendimiento de MCJ2K, usando OSLA y ESLA (tomando como referencia MJ2K).

Observe que las ordenaciones que resuelven ambos algoritmos es similar, lo que significa que aunque el proceso MCTF utilizado por MCJ2K no es ortogonal, puede hacerse una predicción razonable del impacto R/D de las SLs usando ESLA.

Considerando que ESLA tiene una complejidad computacional insignificante y que sólo necesita ser ejecutado en el lado del cliente, ESLA se propone como nuestro algoritmo de rate-allocation usado por defecto en MCJ2K.

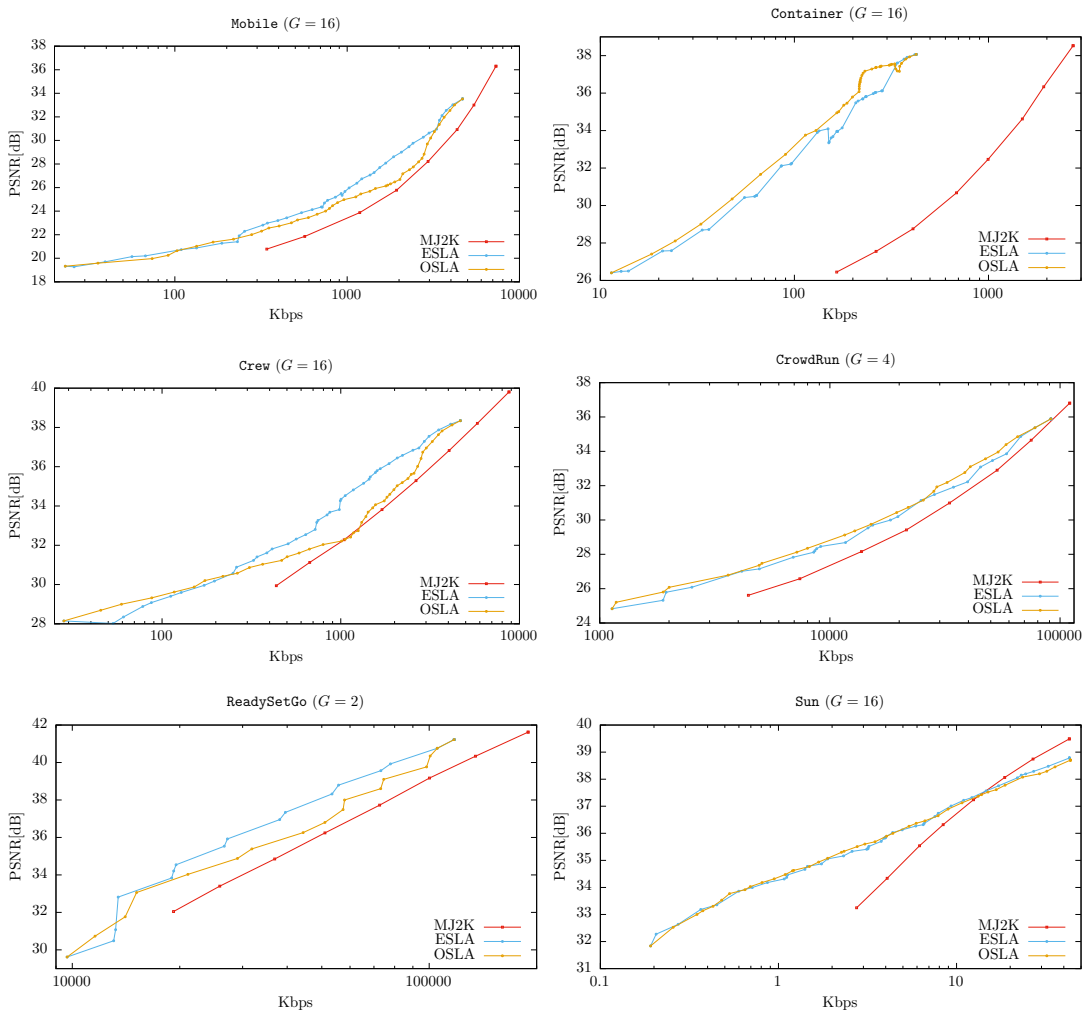


Figura 3.9: MCJ2K (usando OSLA vs. ESLA) vs. MJ2K, para diferentes secuencias.

3.5.2 Acceso aleatorio en MCJ2K

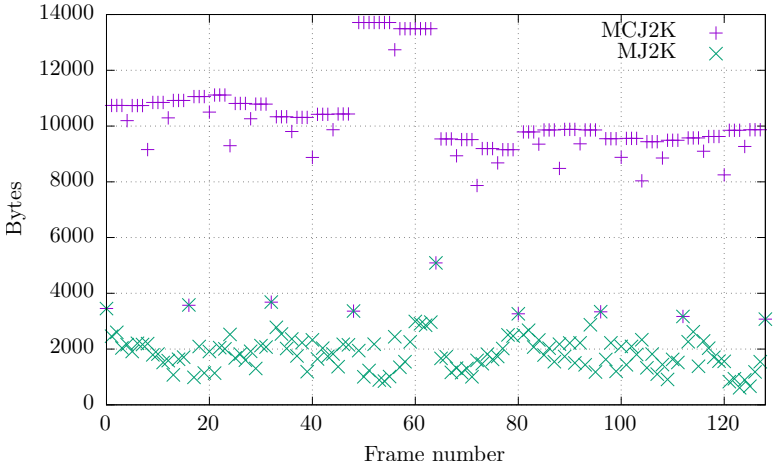
MJ2K no realiza ninguna técnica de correlación temporal entre imágenes. De hecho, codifica cada imagen de la secuencia de forma completamente independiente. MCJ2K obtiene un rendimiento superior de R/D porque encuentra y reduce esta correlación temporal, al igual que el resto de códecs con estimación de movimiento. Por ejemplo, SHVC codifica algunas imágenes a partir de la codificación de otras adyacentes (en caso de GOPs abiertos incluso de otros GOPs).

La ventaja es clara, un rendimiento R/D mejorado cuando se hace streaming de un GOP. Pero en un escenario en el que el cliente sólo solicita una única imagen, los códecs con compensación de movimiento necesitan información de otras imágenes para reconstruir la solicitada. Por lo tanto, para descodificar sólo una imagen B dada, es necesario descomprimir diferentes subbandas (véase la Fig. 1.9), lo que implica un gran coste adicional de bit-rate. Teniendo en cuenta que la mayoría de las imágenes en un GOP suelen ser de este tipo, es un coste considerable en escenarios de acceso aleatorio.

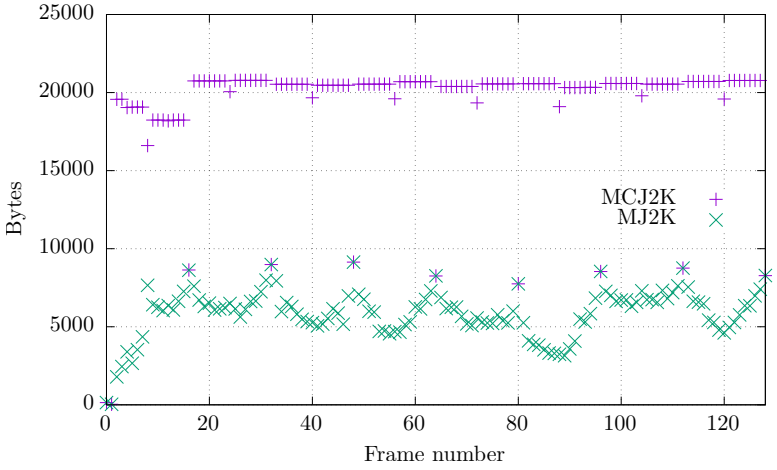
En la Fig. 3.10 se muestran los bytes necesarios para la reconstrucción de cada imagen de la secuencia de forma individual, es decir, en un escenario en el que el cliente sólo solicita una única imagen. Esta relación entre la imagen y el número de bytes necesario para su reconstrucción se muestra para un códec con compensación de movimiento (MCJ2K) y otro sin ella (MJ2K). Por cada vídeo, se han comprimido y codificado 8 GOPs, 16 imágenes por GOP, es decir, 5 TRLs. En MCJ2K se ha seleccionado un nivel de cuantificación para que la distorsión (medida en PSNR) media de la reconstrucción del vídeo sea de 40 dB (un valor típico cualquiera). Tras esto, se ha creado otro code-stream codificando imagen a imagen con MJ2K, variando la cuantificación para igualar la distorsión, imagen a imagen entre ambos códecs.

Observe que ambos códecs necesitan el mismo número de bytes para codificar las imágenes I de cada GOP. El resto de las imágenes necesitan un mayor ancho de banda en MCJ2K para ser transmitidas, debido al aumento de imágenes que deben ser descodificados junto a los correspondientes campos de movimiento, para obtener la imagen solicitada. De este experimento, podemos concluir que:

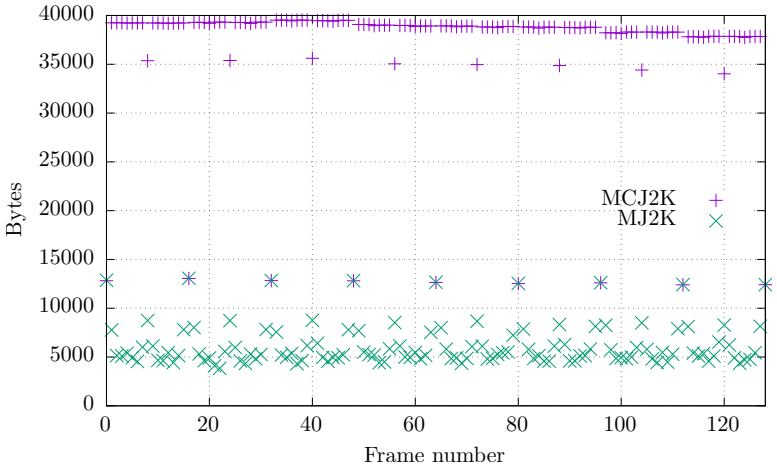
1. El acceso aleatorio a una imagen utilizando MCTF provoca un coste de $400\,\%$ en promedio para reconstrucciones a alta calidad (40 dB).



City



Parkrun



Crowdrun

Figura 3.10: Bytes necesarios para la reconstrucción de una única imagen en MCJ2K vs. MJ2K

2. El acceso secuencial a varias imágenes del vídeo reduce la diferencia entre MCJ2K y MJ2K. Cuando el cliente solicita al menos dos imágenes secuenciales (adyacentes), la ventaja de MJ2K disminuye drásticamente, siendo entonces el coste de MCJ2K de un $25\,\%$. El coste de MCJ2K continua disminuyendo al aumentar el número de imágenes adyacentes solicitadas según cada vídeo.

3.5.3 MCJ2K vs. otros códecs de vídeo

La Fig. 3.11 muestra el rendimiento de compresión de MCJ2K junto a otros códecs de vídeo estándar (SHVC, HEVC, AVC y MPEG-2). MCJ2K usa como técnica de rate-allocation ESLA y los parámetros de compresión optimizados encontrados en experimentos anteriores. Las curvas dibujadas con líneas discontinuas representan una descodificación de code-streams no escalables, y las líneas continuas una descodificación progresiva de code-streams escalables. Los códecs de vídeo no escalables generalmente producen code-streams cuyas reconstrucciones tienen mejor relación R/D que los códecs de vídeo escalables.

En términos generales al comparar MCJ2K con el resto de códec en la comparativa se obtiene la siguiente información.

1. Respecto de AVC y HEVC, MCJ2K necesita aproximadamente un $50\,\%$ más de bit-rate para conseguir la misma calidad en la reconstrucción, pero esta diferencia decrece hasta hacerse incluso nula en algunos casos de estudio.

2. Comparado con SHVC, MCJ2K tiene un rendimiento R/D competitivo. Llegando a presentar incluso mejores resultados en algunos casos de estudio. Nótese que MCJ2K tiene actualmente una predicción de movimiento básica.

   Es destacable la mayor granularidad de MCJ2K respecto de SHVC, que le permite en algunos casos comenzar el streaming de vídeo con un ancho de banda menor.

3. En relación a MPEG-2, que se trata de un códec que implementa un esquema de MCTF similar al usado en MCJ2K, los resultados en términos R/D de MCJ2K son consistentemente mejores.

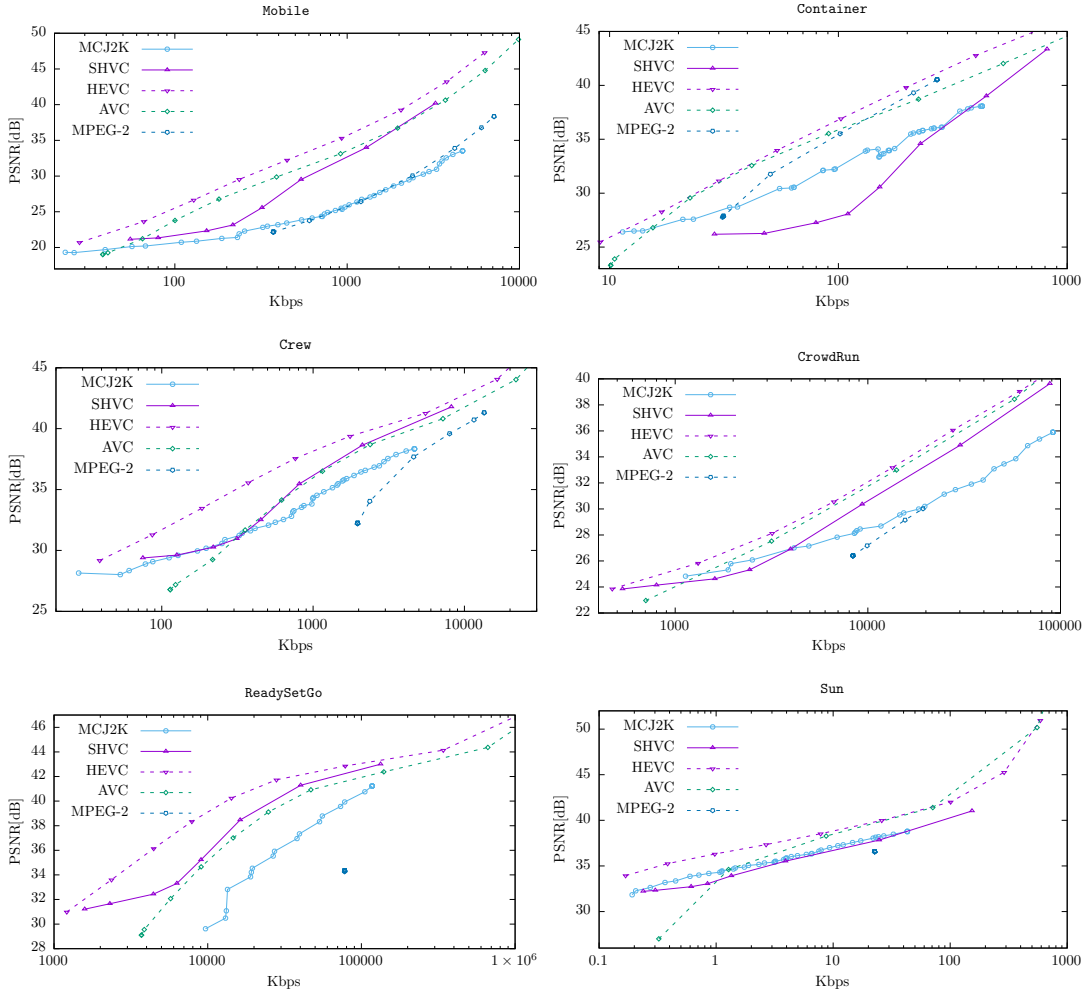


Figura 3.11: MCJ2K (usando ESLA) vs. otros códecs, para diferentes secuencias.

A la vista de la comparativa con los resultados obtenidos por otros códecs de vídeo escalables (H.26*/SVC) y no escalables (H.26*/AVC). Los resultados experimentales muestran que MCJ2K es competitivo, especialmente para la tarea de compresión de secuencias de imágenes de alta resolución. Incluso usando un ME/MC básico y sin disponer de una técnica óptima de rate-allocation entre las imágenes de una subbanda temporal, ni entre las propias subbandas temporales.

Capítulo

4

Conclusiones

Este trabajo presenta MCJ2K, una extensión directa (compatible con JPIP) del estándar MJ2K que puede utilizarse para explotar la redundancia temporal de las secuencias de imágenes. MCJ2K ha sido evaluado con dos algoritmos diferentes de rate-allocation (OSLA y ESLA) propuestos para ser utilizados en un escenario de streaming en el que se utiliza la escalabilidad en calidad. Tras analizar su desempeño, se puede concluir lo siguiente:

1. El ratio de compresión obtenido por MCJ2K es superior al de MJ2K en la mayoría de los vídeos evaluados. La compensación de movimiento implementada mejora entre un 5 % y un 25 % el R/D, proporcionalmente a la predictibilidad del movimiento respecto de MJ2K.

2. OSLA calcula la mejor SL a añadir a un code-stream, pero sin optimizar el resto de la curva R/D.

3. El rendimiento R/D de la estimación que realiza ESLA depende del grado de ortogonalidad del proceso MCTF para la codificación y descodificación de un vídeo dado.

4. OSLA y ESLA siempre mejoran el rendimiento R/D de MCJ2K respecto de no usarlos. Dado que ambos tienen resultados similares y ESLA tiene una complejidad computacional insignificante, ESLA es adecuado para el rate-allocation de MCJ2K en escenarios de streaming en tiempo real.

5. A pesar de que la ME/MC de MCJ2K es básica comparada con la usada por SHVC, el ratio de compresión de MCJ2K es competitivo con éste cuando las

características de movimiento en el vídeo pueden ser predichas por nuestro estimador de movimiento, especialmente a tasas de bits muy bajas.

6. En MCJ2K el tamaño de GOP afecta significativamente al rendimiento R/D. Dicho tamaño debe tener una valor alto si la correlación temporal del vídeo puede ser reducida eficientemente con nuestro estimador de movimiento, y menor en caso contrario.

7. El rango de bit-rate en el que un code-stream MCJ2K puede ser descomprimido suele ser más amplio que en SHVC, así MCJ2K en algunos casos puede comenzar el streaming partiendo de un ancho de banda menor que SHVC. En comparación con MJ2K, MCJ2K necesita de 3 a 50 veces menos bit-rate para reconstruir el GOP a mínima calidad.

8. MCJ2K tiene mayor granularidad que SHVC, es decir, mayor número de puntos de truncado.

Capítulo

5

Trabajo futuro

Las principales líneas de trabajo que se derivan de la presente tesis son:

1. Un code-stream MCJ2K es totalmente compatible con cualquier servidor JPIP estándar, ya que todo el code-stream MCJ2K puede ser almacenado en archivos MJ2K estándar. La adaptación de MCJ2K a servicios JPIP estándar, como Helioviewer, sería una contribución novedosa. MCJ2K mejoraría los sistemas de streaming JPIP al explotar la redundancia temporal de la mayoría de las secuencias de imágenes durante la compresión.

   En un entorno cliente/servidor, el servidor JPIP estándar enviaría, cada subbanda temporal (junto a sus MVs si los tiene) como secuencias MJ2K estándar. El cliente JPIP, para reproducir un stream de MCJ2K, solicitaría un número determinado de capas de calidad de una subbanda temporal (representada mediante un code-stream MJ2K). Éstas deben ser descomprimidas (lo cual también es una operación estándar) y después se ejecutaría una etapa MCTF inversa para renderizar el vídeo.

2. Iterar OSLA para una búsqueda exhaustiva de un conjunto SLs con mayor pendiente hasta llegar a un bit-rate dado.

3. Encontrar una representación escalable en calidad de los datos de movimiento, evitando imprecisiones en los MVs que descoloquen macro-bloques y aumenten la distorsión, en lugar reducirla. Esta contribución permitiría un mayor rate-control sobre los MVs.

4. Estudiar el uso de esquemas MCTF más precisos debería aumentar los ratios de compresión. Por ejemplo: tamaño de macro-bloques y número de TRLs variable;

búsqueda de bloques en traslación, rotación o escala.

5. Analizar el uso de esquemas de codificación en los que la información de movimiento pueda ser estimada en el descodificador (para evitar ser enviada como parte del code-stream). Esto puede llevarse a cabo en aquellos contextos en los que el movimiento a gran escala es predecible, como en las secuencias de imágenes del Sol, cuya velocidad de rotación es estable y conocida.

6. Estudiar cómo afecta MCJ2K a la escalabilidad espacial/ROI proporcionada por el estándar J2K. La dependencia entre imágenes provocada por la explotación de la correlación temporal puede restringir las escalabilidades del code-stream.

Apéndice

A

Publicaciones surgidas de esta tesis

El trabajo de investigación realizado para la presente tesis dio lugar a varias publicaciones. En el presente apéndice se enumeran las mismas junto con sus respectivos indicadores de calidad y clasificadas por su año de publicación (la más reciente primero), dentro de cada categoría.

A.1 Publicaciones en revistas internacionales

[25] J.C. Maturana-Espinosa, D.M., J.P. García-Ortiz & Gonzalez-Ruiz, V. (2019). Layer selection in progressive transmission of motion-compensated jpeg2000 video. *Electronics*, DOI: 10.3390/electronics8091032. **8** Impact factor JCR 2018: 1.764. JCR category rank: 154/265 (Q3) in *Engineering, Electrical Electronic*.

A.2 Publicaciones en proceedings de conferencias internacionales

[24] J.C. Maturana-Espinosa, D.M., J.P. García-Ortiz & Gonzalez-Ruiz, V. (2018). Rate allocation for motion compensated jpeg2000. DOI: 10.1109/DCC.2018.00015. *IEEE 2018 Data Compression Conference (DCC)*.

A.3 Publicaciones en conferencias nacionales

[33] Maturana-Espinosa, C., García-Sobrino, J., Sánchez-Hernández, J. & González-Ruiz, V. (2013). Codificación de vídeo escalable usando motion compensated jpeg2000 con control de bit-rate. In *Actas de las XXIV Jornadas de Paralelismo*, 324–329, Sociedad de Arquitectura y Tecnología de Computadores (SARTECO). Universidad Complutense de Madrid, Servicio de Publicaciones. Universidad Complutense de Madrid, Madrid, Spain

[32] Maturana-Espinosa, C., Sánchez-Hernández, J., García-Ortiz, J. & González-Ruiz, V. (2012). Scalable video coding using motion compensated jpeg 2000. In *Proceedings of the II Workshop on Multimedia Data Coding and Transmission (WMDCT)*, 38–43, Grupo de Arquitectura y Tecnología de Computadores (GATCOM), Universidad Miguel Hernández, Elche, Alicante, Spain

Apéndice

B

Otras publicaciones producidas durante la elaboración de esta Tesis

El esfuerzo de investigación invertido durante el periodo de tiempo en el que se elaboró esta tesis produjo algunas publicaciones adicionales como resultados colaterales no incluidos en este documento.

B.1 Publicaciones en revistas nacionales

[49] Sánchez-Hernández, J., García-Ortiz, J., Maturana-Espinosa, J.C., González-Ruiz, V. & Müller, D. (2013). Streaming interactivo de secuencias de imágenes jpeg2000 de alta resolución. In *Actas de las XXIV Jornadas de Paralelismo*, 139–144, Sociedad de Arquitectura y Tecnología de Computadores (SARTECO). Universidad Complutense de Madrid, Servicio de Publicaciones. Universidad Complutense de Madrid, Madrid, Spain

[48] Sánchez-Hernández, J., García-Ortiz, J., Martín, C., Maturana-Espinosa, C., González-Ruiz, V. & Müller, D. (2012). Improved jpip-compatible architecture for video streaming of jpeg 2000 image sequences. In *Proceedings of the II Workshop on Multimedia Data Coding and Transmission (WMDCT)*, 33–37, Grupo de Arquitectura y Tecnología de Computadores (GATCOM), Universidad Miguel Hernández, Elche, Alicante, Spain

Bibliografía

[1] Andre, T., Cagnazzo, M., Antonini, M. & Barlaud, M. (2007). JPEG2000-compatible scalable scheme for wavelet-based video coding. *EURASIP Journal on Image and Video Processing*, **2007**, 1–11, doi:10.1155/2007/30852.

[2] Andreopoulos, Y., van der Schaar, M., Munteanu, A., Barbarien, J., Schelkens, P. & Cornelis, J. (2003). Fully-Scalable Wavelet Video Coding Using In-Band Motion Compensated Temporal Filtering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 3, 417 – 420.

[3] Andreopoulos, Y., van der Schaar, M., Munteanu, A., Barbarien, J., Schelkens, P. & Cornelis, J. (2003). Fully-scalable wavelet video coding using in-band motion compensated temporal filtering. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 3, 417–420, IEEE.

[4] Aulí-Llinàs, F., Bilgin, A. & Marcellin, M. (2011). FAST Rate Allocation Through Steepest Descent for JPEG2000 video transmission. *Image Processing, IEEE Transactions on*, DOI: 10.1109/TIP.2010.2077304. **20**, 1166–1173.

[5] Avramova, Z., Vleeschauwer, D.D., Spaey, K., Wittevrongel, S., Bruneel, H. & Blondia, C. (2007). Comparison of simulcast and scalable video coding in terms of the required capacity in an IPTV network. In *Proceedings of the Packet Video Conference*, 113 – 122, Laussane, Switzerland.

[6] Barbarien, J., Munteanu, A., Verdicchio, F., Andreopoulos, Y., Cornelis, J. & Schelkens, P. (2005). Motion and texture rate-allocation for prediction-based scalable motion-vector coding. *Signal Processing: Image Communication*, **20**, 315–342.

[7] Bilgin, A. & Marcellin, M. (2006). JPEG2000 for digital cinema. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, DOI: 10.1109/ISCAS.2006.1693475. 4 pp. – 3881.

[8] Boisson, G. & François, E. (2006). Removing redundancy in multi-resolution scalable video coding schemes. In *Proceedings of 13th IEEE International Conference on Image Processing, ICIP'2006*, Atlanta, GA.

[9] Boisson, G., François, E. & Guillemot, C. (2004). Accuracy scalable motion coding for efficient scalable video compression. In *Proceedings of 11th IEEE International Conference on Image Processing, ICIP'2004*, Singapore.

[10] Boyce, J.M., Ye, Y., Chen, J. & Ramasubramonian, A.K. (2016). Overview of shvc: scalable extensions of the high efficiency video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, **26**, 20–34.

[11] C. Christopoulos, A.S. & Ebrahimi, T. (2000). The jpeg2000 still image coding system: An overview. *IEEE Transactions on Consumer Electronics*, **46**, 1103–1127.

[12] Cagnazzo, M., Castaldo, F., Andre, T., Antonini, M. & Barlaud, M. (2007). Optimal motion estimation for wavelet motion compensated video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, DOI: 10.1109/TCSVT.2007.897110. **17**, 907–911.

[13] Choi, S.J. & Woods, J.W. (1999). Motion-compensated 3-d subband coding of video. *IEEE Transactions on Image Processing*, **8**, 155–167.

[14] Cohen, R. & Woods, J. (2007). Resolution scalable motion-compensated jpeg 2000. In *2007 15th International Conference on Digital Signal Processing*, 459–462, IEEE.

[15] Dagher, J., Bilgin, A. & Marcellin, M. (2003). Resource-constrained rate control for Motion JPEG2000. *Image Processing, IEEE Transactions on*, DOI: 10.1109/TIP.2003.819228. **12**, 1522–1529.

[16] Fowler, J., Cui, S. & Wang, Y. (2006). Motion compensation via redundant-wavelet multihypothesis. *IEEE Transactions on Image Processing*, **15**, 3102–3113.

[17] Gibson, D. & Spann, M. (1999). Multi-frame motion estimation: Application to motion compensated prediction. In *Proceedings of the 1999 IEEE International Symposium onCircuits and Systems, ISCAS 99*, vol. 4, 54–57.

[18] Goyal, V.K. (2001). Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, 74 – 93.

[19] Hsieh, C. & Liu, Y. (2000). Fast search algorihtms for vector quantization of images using multiple triangle inequalities and wavelet transform. *IEEE Trans. on Image Proc.*, **9**, 321–328.

[20] International Organization for Standardization (2004). Information Technology - JPEG 2000 Image Coding System - Part 3: Motion JPEG 2000.

[21] ISO (2004). Information Technology - JPEG 2000 Image Coding System - Core Coding System. ISO/IEC 15444-1:2004.

[22] ISO (2007). Information Technology - JPEG 2000 Image Coding System: Motion JPEG 2000. ISO/IEC 15444-3:2007.

[23] ITU (2005). Information Technology - JPEG 2000 Image Coding System: Interactivity Tools, APIs and Protocols. http://www.itu.int/rec/T-REC-T.808-200501-I.

[24] J.C. Maturana-Espinosa, D.M., J.P. García-Ortiz & Gonzalez-Ruiz, V. (2018). Rate allocation for motion compensated jpeg2000. DOI: 10.1109/DCC.2018.00015. *IEEE 2018 Data Compression Conference (DCC)*.

[25] J.C. Maturana-Espinosa, D.M., J.P. García-Ortiz & Gonzalez-Ruiz, V. (2019). Layer selection in progressive transmission of motion-compensated jpeg2000 video. *Electronics*, DOI: 10.3390/electronics8091032. **8**.

[26] Jiménez-Rodríguez, L., Aulí-Llinàs, F. & Marcellin, M. (2013). FAST rate allocation for JPEG2000 video transmission over time-varying channels. *IEEE Transactions on Multimedia*, DOI: http://dx.doi.org/10.1109/TMM.2012.2199973. **15**, 15–26.

[27] Kakadu Software Pty Ltd based in Sydney, Australia (2018). Kakadu JPEG 2000 SDK. http://www.kakadusoftware.com.

[28] Karlsson, G. & Vetterli, M. (1988). Three-dimensional subband coding of video. In *International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1100–1103.

[29] Luo, L., Li, J., Li, S., Zhuang, Z. & Zhang, Y.Q. (2001). Motion compensated lifting wavelet and its application in video coding. In *IEEE International Conference on Multimedia and Expo*, 365–368.

[30] M. Ferroukhi, M.A.Y.H., A. Ouahabi & Taleb-Ahmed, A. (2019). Medical video coding based on 2nd-generation wavelets:performance evaluation. *Electronics*, **8(1)**.

[31] Maani, E. & Katsaggelos, A. (2010). Unequal error protection for robust streaming of scalable video over packet lossy networks. *IEEE Transactions on Circuits and Systems for Video Technology*, **20**, 407 – 416.

[32] Maturana-Espinosa, C., Sánchez-Hernández, J., García-Ortiz, J. & González-Ruiz, V. (2012). Scalable video coding using motion compensated jpeg 2000. In *Proceedings of the II Workshop on Multimedia Data Coding and Transmission (WMDCT)*, 38–43, Grupo de Arquitectura y Tecnología de Computadores (GATCOM), Universidad Miguel Hernández, Elche, Alicante, Spain.

[33] Maturana-Espinosa, C., García-Sobrino, J., Sánchez-Hernández, J. & González-Ruiz, V. (2013). Codificación de vídeo escalable usando motion compensated jpeg2000 con control de bit-rate. In *Actas de las XXIV Jornadas de Paralelismo*, 324–329, Sociedad de Arquitectura y Tecnología de Computadores (SARTECO). Universidad Complutense de Madrid, Servicio de Publicaciones. Universidad Complutense de Madrid, Madrid, Spain.

[34] Mehrotra, S. & Zhao, W. (2009). Rate-distortion optimized client side rate control for adaptive media streaming. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, DOI: 10.1109/MMSP.2009.5293246. 1–6.

[35] Mehrseresht, N. & Taubman, D. (2004). An efficient content-adaptive mc 3d-dwt with enhanced spatial and temporal scalability. In *International Conference on Image Processing, ICIP 04*, vol. 2, 1329–1332.

[36] Mehrseresht, N. & Taubman, D. (2004). A flexible structure for fully scalable motion compensated 3d-dwt with emphasis on the impact of spatial scalability. *IEEE Transactions on Image Processing*, **15**, 740–753.

[37] Mokry, R. & Anastassiou, D. (1994). Minimal error drift in frequency scalability for motion-compensated DCT coding. *IEEE Trans. Circuits Syst. Video Technol.*, **4**, 392 –406.

[38] Müller, D., Fleck, B., Dimitoglou, G., Caplins, B.W., Amadigwe, D.E., Ortiz, J.P.G., B. Wamsler, A.A., Hughitt, V.K. & Ireland, J. (2009). JHelioviewer: Visualizing large sets of solar images using JPEG 2000. *Computing in Science and Engineering*, **11**, 38–47.

[39] Müller, D., Nicula, B., Felix, S., Verstringe, F., Bourgoignie, B., Csillaghy, A., Berghmans, D., Jiggens, P., García-Ortiz, J., Ireland, J., Zahniy, S. & Fleck, B. (2017). JHelioviewer-Time-dependent 3D visualisation of solar and heliospheric data. *Astronomy & Astrophysics*, **606**, A10.

[40] Munteanu, A., Andreopoulos, Y., van der Schaar, M., Schelkens, P. & Cornelis, J. (2003). Control of the distortion variation in video coding systems based on motion compensated temporal filtering. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 2, DOI: 10.1109/ICIP.2003.1246616. II – 61–4 vol.3.

[41] Naman, A. & Taubman, D. (2011). JPEG2000-based Scalable Interactive Video (JSIV). *Image Processing, IEEE Transactions on*, DOI: http://dx.doi.org/10.1109/TIP.2010.2093905. **20**, 1435– 1449.

[42] Naman, A. & Taubman, D. (2011). Jpeg2000-based scalable interactive video (jsiv) with motion compensation. *Image Processing, IEEE Transactions on*, DOI: 10.1109/TIP.2011.2126588. **20**, 2650 –2663.

[43] Naman, A. & Taubman, D. (2011). JPEG2000-based Scalable Interactive Video (JSIV) with motion compensation. *Image Processing, IEEE Transactions on*, DOI: 10.1109/TIP.2011.2126588. **20**, 2650 –2663.

[44] Narroschke, M. (2005). Benefits and costs of scalable video coding for internet streaming. *J. Vis. Comun. Image Represent.*, DOI: 10.1016/j.jvcir.2004.11.010. **16**, 397–411.

[45] Ohm, J.R. (1994). Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, **3**, 559–571.

[46] Pesquet-Popescu, B. & Bottreau, V. (2001). Three-dimensional lifting schemes for motion compensated video compression. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 1793–1796.

[47] Ponec, M., Sengupta, S., Chen, M., Li, J. & Chou, P. (2011). Optimizing multirate peer-to-peer video conferencing applications. *IEEE Transactions on Multimedia*, **13**, 856 – 868.

[48] Sánchez-Hernández, J., García-Ortiz, J., Martín, C., Maturana-Espinosa, C., González-Ruiz, V. & Müller, D. (2012). Improved jpip-compatible architecture for video streaming of jpeg 2000 image sequences. In *Proceedings of the II Workshop on Multimedia Data Coding and Transmission (WMDCT)*, 33–37, Grupo de Arquitectura y Tecnología de Computadores (GATCOM), Universidad Miguel Hernández, Elche, Alicante, Spain.

[49] Sánchez-Hernández, J., García-Ortiz, J., Maturana-Espinosa, J.C., González-Ruiz, V. & Müller, D. (2013). Streaming interactivo de secuencias de imágenes jpeg2000 de alta resolución. In *Actas de las XXIV Jornadas de Paralelismo*, 139–144, Sociedad de Arquitectura y Tecnología de Computadores (SARTECO). Universidad Complutense de Madrid, Servicio de Publicaciones. Universidad Complutense de Madrid, Madrid, Spain.

[50] Sánchez-Hernández, J., García-Ortiz, J., González-Ruiz, V. & Müller, D. (2015). Interactive streaming of sequences of high resolution jpeg2000 images. *IEEE Transactions on Multimedia*, **17**, 1829–1838.

[51] Schwarz, H., Marpe, D. & Wiegand, T. (2007). Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 1103–1120.

[52] Schwarz, H., Marpe, D. & Wiegand, T. (2007). Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Transactions on circuits and systems for video technology*, **17**, 1103–1120.

[53] Secker, A. & Taubman, D. (2001). Motion-compensated highly scalable video compression using an adaptive 3D wavelet transform based on lifting. *Proceedings of the IEEE International Conference on Image Processing. Thessaloniki, Greece*, **2**, 1029–1032.

[54] Secker, A. & Taubman, D. (2003). Lifting-based invertible motion adaptive transform (limat) framework for highly scalable video compression. *IEEE Transactions on Image Processing*, **12**, 1530–1542.

[55] Secker, A. & Taubman, D. (2003). Lifting-based Invertible Motion Adaptive Transform (LIMAT) framework for highly scalable video compression. *IEEE Transactions on Image Processing*, DOI: 10.1109/TIP.2003.819433. **12**, 1530–1542.

[56] Suguri, K., Minami, T., Matsuda, H., Kusaba, R., Kondo, T., Kasai, R., Watanabe, T., Sato, H., Shibata, N., Tashiro, Y. *et al.* (1996). A real-time motion estimation and compensation lsi with wide search range for mpeg2 video encoding. *IEEE journal of solid-state circuits*, **31**, 1733–1741.

[57] Sullivan, G., Ohm, J., Han, W.J. & Wiegand, T. (2012). Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, DOI: 10.1109/TCSVT.2012.2221191. **22**, 1649–1668.

[58] Sullivan, G.J. & Wiegand, T. (1998). Rate-distortion optimization for video compression. *IEEE signal processing magazine*, **15**, 74–90.

[59] Sun, H., Vetro, A. & Xin, J. (2007). An overview of scalable video streaming: Research articles. *Wirel. Commun. Mob. Comput.*, DOI: 10.1002/wcm.v7:2. **7**, 159–172.

[60] Sweldens, W. (1995). The Lifting Scheme: A new Philosophy in Biorthogonal Wavelet Constructions. In *Proc. SPIE*, vol. 2569, 68–79.

[61] Taubman, D. & Marcellin, M. (2002). *JPEG2000 image compression fundamentals, standards and practice*. DOI: 10.1007/978-1-4615-0799-4. Springer Science & Business Media, LLC, pg.418.

[62] Technical Committee: ISO/IEC JTC 1/SC 29 (2012). ISO/IEC 23009-1:2012 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623.

[63] van der Schaar, M. & Turaga, D.S. (2003). Unconstrained motion compensated temporal filtering (umctf) framework for wavelet video coding. In *International Conference on Acoustics, Speech, and Signal Processing*, vol. 3.

[64] Vetro, A., Christopoulos, C. & Sun, H. (2003). An overview of video transcoding architectures and techniques. *IEEE Signal Processing Magazine*, **20**, 18 – 29.

[65] Wiegand, T., Sullivan, G.J., Bjøntegaard, G. & Luthra, A. (2003). Overview of the h.264/avc video coding standard. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, **13**.

[66] Wu, S. & Gersho, A. (1993). Joint estimation of forward/backward motion vectors for MPEG interpolative prediction. vol. 1, 14.3.2–14.3.2.

[67] Xiong, R., Xu, J., Wu, F. & Li, S. (2006). Adaptive MCTF based on correlation noise model for SNR scalable video coding. In *IEEE International Conference on Multimedia and Expo (ICME)*, 1865–1868.

[68] Yasuda, H. & Kawanishi, H. (1978). Predictor Adaptive DPCM. *SPIE Conference on Applications of Digital Image Processing*, **149**, 189–195.

[69] Zhang, A., Deering, S., Estrin, D., Shenker, S. & Zappala, D. (1993). RSVP: a new resource ReSerVation Protocol. *IEEE Network*, **7**, 8 – 18.

[70] Zou, J., Xiong, H., Li, C., Song, L., He, Z. & Chen, T. (2011). Prioritized flow optimization with multi-path and network coding based routing for scalable multirate multicasting. *IEEE Transactions on Circuits and Systems for Video Technology*, **21**, 259 – 273.