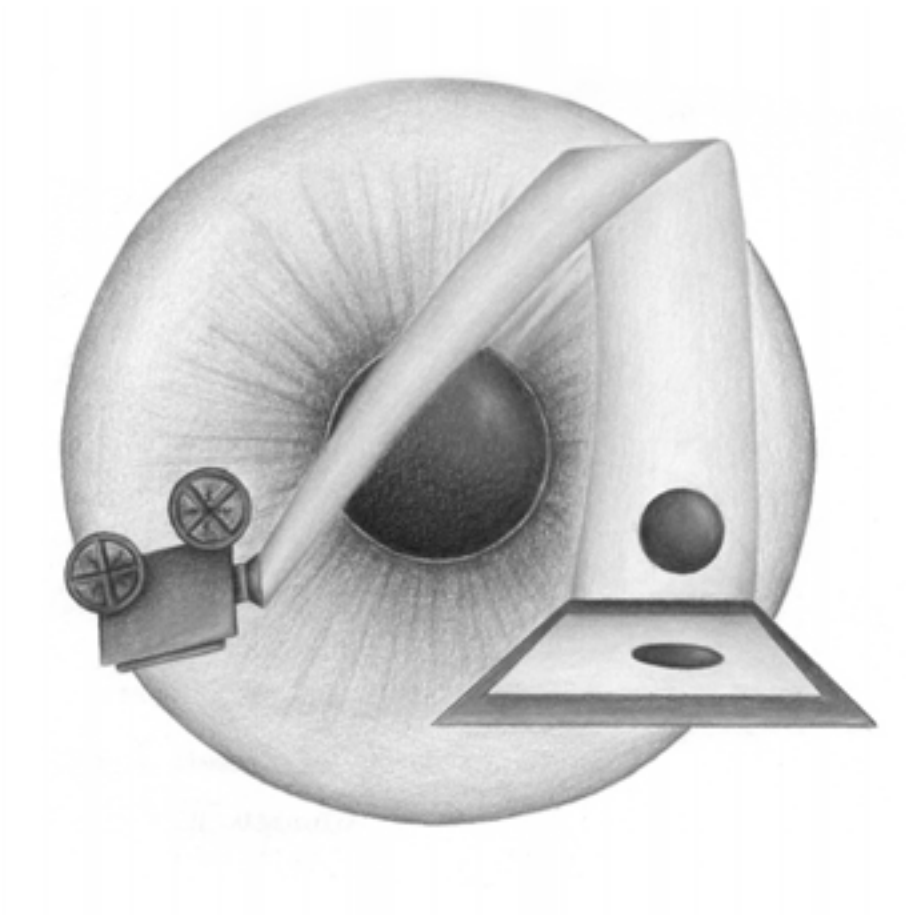
Estándares de compresión de imágenes fijas



Índice

Etapa 1: La necesidad de compresión

Introducción

Una parte considerable del volumen de información asociado al tratamiento de datos en aplicaciones multimedia se origina en la transmisión o almacenamiento de fotografías y señales de audio y vídeo. Las fotografías son un recurso habitual en páginas web, aplicaciones de acceso a datos (enciclopedias), publicidad, etc. La cantidad de información (bits) asociada a una fotografía depende de su resolución (tamaño) y su calidad (bits por píxel). En general, para resoluciones convencionales, el flujo de datos asociado a una fotografía es notablemente superior al que requieren las partes de texto, gráficos y animaciones. Las señales de audio y vídeo representan un problema parecido en lo que se refiere al volumen de información y serán consideradas con detalle en otros módulos.

A finales de la década de los ochenta, se popularizaron distintas aplicaciones informáticas que permitían el tratamiento y la digitalización de imágenes procedentes de una cámara o un escáner. Las empresas desarrolladoras definieron distintos formatos propietarios para el almacenamiento de las imágenes. Estos formatos eran totalmente incompatibles y para realizar el intercambio entre distintas aplicaciones era necesario que cada desarrollador proporcionara rutinas específicas para abrir el resto de los formatos. En algunos casos se utilizaban algoritmos de compresión de imagen para reducir el espacio de disco que ocupaban las imágenes, pero, en otros, la incompatibilidad se debía exclusivamente al uso de cabeceras propietarias y ordenaciones distintas de los píxeles en la imagen. Los formatos que tuvieron un mayor impacto comercial fueron BMP, PIC, PCX, TRT, RAW, TIF, GIF, BIG, etc., algunos de los cuales todavía se utilizan actualmente.

El comité JPEG

A mediados de los ochenta, la ISO/IEC fomentó la creación de un grupo de expertos para que trabajase de forma coordinada en un estándar para el almacenamiento y transmisión de imágenes comprimidas. Este grupo se denominó Joint Photographic Experts Group (JPEG), nombre que posteriormente se utilizó como nombre del formato de compresión. Inicialmente, en marzo de 1987, se propuso un total de estrategias distintas para la compresión de imágenes. Estas propuestas fueron valoradas cuidadosamente hasta que finalmente se acordó utilizar un esquema de compresión basado en el uso de una transformada coseno adaptada al contenido de la imagen. Uno de los objetivos principales era obtener un estándar que permitiera obtener calidades de imagen aceptables para la transmisión a través de líneas digitales de baja velocidad (64 kbps) utilizando un *hardware* de bajo coste.

El resultado de este trabajo fue el estándar JPEG cuya versión básica apareció en junio de 1991 y cuyo uso se ha popularizado durante la década de los noventa como formato de intercambio entre distintas aplicaciones y que ha desplazado completamente a los formatos propietarios de la década de los ochenta. Desde la primera aparición del estándar, el comité ha seguido proponiendo continuas enmiendas y mejoras del procedimiento de compresión básico. Recientemente se ha introducido un nuevo estándar, conocido como el JPEG2000, que mejora considerablemente las prestaciones del primero y que se supone que a medio plazo sustituirá completamente al estándar original.

El JPEG clásico combina distintas técnicas de compresión de datos: la transformada coseno, la recuantificación de los datos, Run Length Encoding (codificación por longitud de cadenas), códigos de longitud variable (Huffman), etc. Los procedimientos de compresión están optimizados para la codificación de imágenes naturales multinivel (varios niveles de gris o imágenes en color). Sus prestaciones son excelentes cuando la imagen corresponde a fotografías de escenas naturales, pero presenta varios problemas cuando se incorporan gráficos sintéticos o texto en las imágenes. El JPEG2000 resuelve estas limitaciones y proporciona una capacidad de compresión superior para imágenes naturales.

La imagen sin comprimir

Representación de la imagen mediante componentes RGB

Las imágenes naturales están representadas como matrices que proporcionan información sobre el color de cada elemento de imagen (píxel). Generalmente, cada elemento de imagen contiene 8 bits de información para cada componente de color R, G, B. Esto significa que cada píxel tiene asignado un total de $8 \times 3 = 24$ bits. Con ello, el número de colores que pueden representarse es de:

$$2^{24} = 16.777.216 \text{ posibles colores}$$

Algunos sistemas de representación de imágenes en PC reducen el número de colores representando cada píxel con 16 bits. En este caso, se utilizan 5 bits para representar las componentes roja (R) y azul (B) y 6 bits para la componente verde (G). La mayor sensibilidad del sistema visual humano a las longitudes de onda próximas al verde justifica el uso de un mayor número de bits para esta componente. En este sistema de representación puede representarse un total de:

$$32 \text{ niveles de rojo} \times 64 \text{ niveles de verde} \times 32 \text{ niveles de azul} = 65.536 \text{ colores}$$

Aunque la representación de cada elemento de imagen con 16 bits significa una reducción en el volumen total de datos asignados a la imagen, no suele considerarse como un método de compresión, sino como una forma alternativa de representar la imagen original con menor calidad.

Una representación alternativa es la de 32 bits por píxel. En este caso, se utilizan 8 bits para cada una de las componentes de color RGB (igual que en el caso de la representación con 24 bits) y se añade una componente adicional de 8 bits para representar la transparencia. El factor de transparencia se utiliza para indicar el grado de fusión de una imagen con el fondo.

La principal razón por la que en algunos casos se representan los píxeles utilizando 16 bits o 32 bits en vez de 24 es que las palabras de memoria en un ordenador son de 32 bits. Cuando se utilizan 16 bits por cada píxel, pueden almacenarse 2 elementos de imagen en una misma posición de memoria. Para representaciones con 32 bits, cada píxel está representado en una palabra de memoria. En ambos casos resulta

trivial determinar la posición de memoria en la que está almacenado un determinado píxel, lo que simplifica y acelera los cálculos cuando se aplica algún algoritmo de tratamiento de imagen. En cambio, cuando utilizamos 24 bits, el mapeado de la imagen en la memoria del ordenador es más complejo y las operaciones de direccionamiento de los elementos de imagen suelen incrementar el tiempo de proceso. En la figura siguiente se muestra la distribución de las componentes de color en la memoria del ordenador para los espacios de representación RGB más comunes.

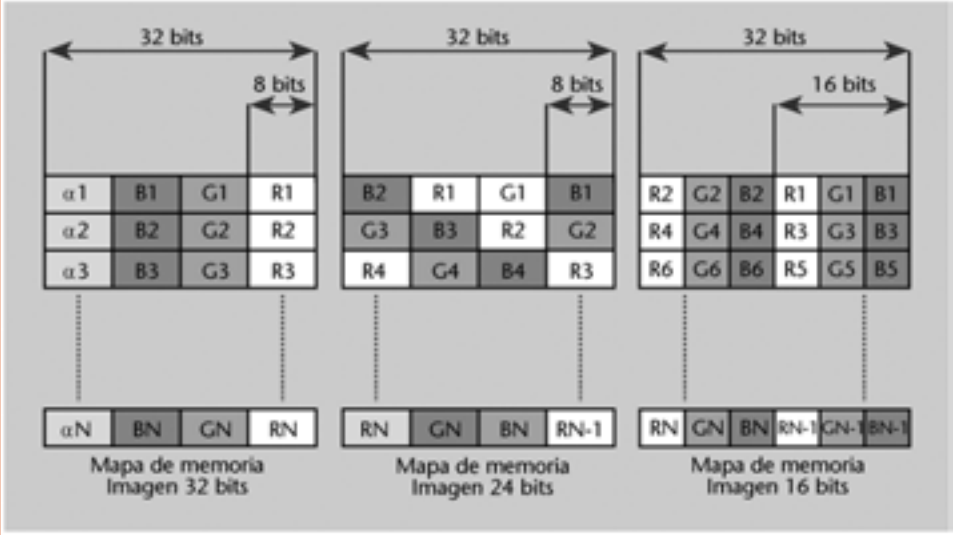


Figura 1. Distribución del mapa de bits de la imagen en memoria

Representación de la imagen mediante componentes YCbCr

Las componentes YCbCr son una forma alternativa de representar la imagen en formato digital. La ventaja principal de estas componentes es que la información de luminancia (Y) y la información de color o croma (Cb y Cr) están en canales separados. Debido a que el sistema visual humano presenta una mejor resolución espacial a la información de luminancia que a la información de croma, es posible reducir el número de muestras de las componentes Cb y Cr sin que aparentemente se produzca una pérdida de calidad.

Las señales YCbCr son una versión escalada y con componente continua añadida de las señales YUV que se utilizan en el estándar analógico del PAL. En el formato digital, la señal de luminancia tiene un margen dinámico nominal que va desde los niveles 16 a 235 (cuantificada en 8 bits) donde 16 corresponde al negro y 235, al blanco. Las señales de croma, también denominadas *diferencia de color*, pueden tomar valores entre 16 y 240, cuantificadas con 8 bits, donde el nivel 128 se toma como la referencia cero. Esto significa que los números menores de 128 representan niveles de croma negativos y viceversa.

Las componentes YCrCb pueden determinarse a partir de las componentes RGB utilizando la siguiente relación algebraica:

$$\begin{aligned} Y' &= 0.299R' + 0.587G' + 0.114B' \\ (B' - Y') &= -0.299R' - 0.587G' + 0.886B' \\ (R' - Y') &= 0.701R' - 0.587G' - 0.114B' \end{aligned}$$

que se normaliza a los valores YCbCr mediante:

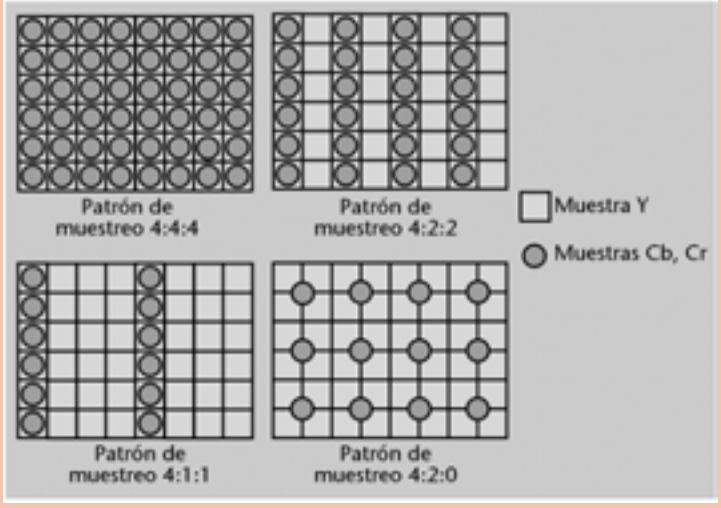
$$\begin{aligned} Y &= 219Y' + 16 \\ Cb &= 224[0.564(B' - Y')] + 128 = 126(B' - Y') + 128 \\ Cr &= 224[0.713(R' - Y')] + 128 = 160(R' - Y') + 128 \end{aligned}$$

La imagen puede almacenarse utilizando las componentes YCbCr y posteriormente recalcular las componentes RGB para representarla en la pantalla del ordenador. Las relaciones para determinar las componentes RGB a partir de las YCbCr pueden obtenerse invirtiendo las ecuaciones anteriores.

Formatos de representación con componentes YCbCr

Cuando se utilizan las componentes YCbCr, la imagen puede representarse mediante los formatos 4:4:4, 4:2:2, 4:1:1 ó 4:2:0. Estos formatos fueron analizados en detalle en la primera parte de la asignatura.

La diferencia más significativa es que en el formato 4:4:4 se dispone de las tres componentes Y, Cb y Cr para cada uno de los elementos de imagen, mientras que en los otros formatos las componentes de croma están submuestreadas. En el 4:2:2 y 4:1:1 sólo se submuestrea en la dirección horizontal (dentro de una línea), mientras que en el 4:2:0 se submuestrea en las dos direcciones (tanto filas como columnas). En la figura siguiente se muestra un esquema de los patrones de submuestreo de cada uno de los formatos.



Patrones de distribución de muestras de luminancia y croma en los principales formatos

Ejemplos de distribución de muestras

A modo de ejemplo, para una imagen de $N \times M$ píxeles (filas × columnas) el número de componentes de luminancia y croma que obtendríamos en cada formato se define en la siguiente tabla.

Formato/tamaño	Y (luminancia)	U (croma)	V (croma)
4:4:4	$N \times M$	$N \times M$	$N \times M$
4:2:2	$N \times M$	$N \times (M/2)$	$N \times (M/2)$
4:1:1	$N \times M$	$N \times (M/4)$	$N \times (M/4)$
4:2:0	$N \times M$	$(N/2) \times (M/2)$	$(N/2) \times (M/2)$

Así, si cada una de las componentes YUV se codifica con 8 bits, el número de bits totales en una imagen de $N \times M$ elementos en el formato 4:4:4 sería:

$$(N \times M) \times 8 \text{ [luminancia]} + (N \times M) \times 8 \text{ [croma U]} + (N \times M) \times 8 \text{ [croma V]} = 24 \times (N \times M) \text{ bits}$$

En cambio, para el formato 4:2:0, el número de bits asociados a una imagen de $N \times M$ elementos viene dado por:

$$(N \times M) \times 8 \text{ [luminancia]} + ((N \times M)/4) \times 8 \text{ [croma U]} + ((N \times M)/4) \times 8 \text{ [croma V]} = 12 \times (N \times M) \text{ bits}$$

Generalmente, los algoritmos de compresión de la imagen se aplican sobre cada una de las componentes de la imagen. En el supuesto de que la imagen original estuviera representada mediante componentes RGB, podemos optar por comprimir cada una de estas tres componentes por separado o convertir la imagen a componentes YCbCr en uno de los formatos anteriores. El JPEG permite codificar un total de 4 componentes por imagen. Los formatos más utilizados por el JPEG son el 4:1:1 y, sobre todo, el 4:2:0, aunque también se admiten el 4:4:4 y el 4:2:2.

Medida del factor de compresión y la calidad de imagen

Para comparar las prestaciones de los algoritmos de compresión se realizan medidas comparativas sobre la cantidad de compresión conseguida y la calidad de imagen. Los parámetros más habituales para cuantificar la compresión son los **bits por píxel** y el **factor de compresión** que se definen en los siguientes subapartados.

La calidad de la imagen suele cuantificarse comparando la imagen comprimida con la original. Una medida habitual es la diferencia o error entre ambas imágenes elevada al cuadrado, que se denomina **error cuadrático**. Otra alternativa es la **relación señal de pico a ruido (PSNR)**.

En ambos casos se trata de medidas cuantitativas que proporcionan una idea global sobre la calidad de la imagen, pero que, en algunos casos, no se corresponden directamente con la calidad visual subjetiva. Estos problemas suelen deberse a la aparición de efectos de bloque o ruido que tienen poca incidencia sobre el error, pero que resultan muy molestos visualmente. Existen diferentes recomendaciones para la medida subjetiva de la calidad de imagen, aunque este tipo de medidas suele utilizarse poco debido a que implica un elevado número de observadores y se trata de medidas lentas y tediosas de realizar.

Compresión de una imagen: bits por píxel

Una forma de medir el grado de compresión de una imagen es mediante el número medio de bits por píxel con el que está codificada. El número de bits por píxel (bpp) se determina como el cociente entre el tamaño total del fichero de imagen (expresado en bits) y el número de píxeles que contiene la imagen (filas × columnas):

bpp (bits por píxel) = Tamaño de la imagen en bits / Número de píxeles

Una imagen original expresada en las componentes RGB con 8 bits por cada componente está representada con 24 bpp. En cambio, una imagen original representada con las componentes YCrCb en el formato 4:2:0 está representada con 12 bpp. Los estándares de compresión JPEG y JPEG2000 permiten representar imágenes con calidades aceptables con menos de 1 bpp.

Compresión de una imagen: factor de compresión

El factor de compresión se define como el cociente entre el tamaño del fichero asociado a la imagen original y el tamaño del fichero de la imagen comprimida. La relación puede expresarse tanto en bits como en *Kbytes*, y se obtiene en ambos casos el mismo resultado:

Factor de compresión = Tamaño fichero original / Tamaño fichero imagen comprimida

El factor de compresión también puede expresarse de forma alternativa como el cociente entre los bits por píxel con los que está codificada la imagen original y los bits por píxel de la imagen comprimida.

Factor de compresión = bpp imagen original / bpp imagen comprimida

El estándar JPEG permite obtener factores de compresión de 15 con una calidad de imagen comparable al original.

Ejemplo de cálculo de factores de compresión y bpp

Supongamos una imagen de 768 × 1024 codificada con componentes RGB (bitmap de 24 bits) que al comprimirla resulta en un tamaño de fichero de 65 *Kbytes*. Se pide calcular el factor de compresión y el bpp de la imagen comprimida.

Para determinar el factor de compresión debemos calcular primero el tamaño del fichero asociado a la imagen original. Suponiendo que pueden despreciarse los bits de cabecera de fichero (dtos auxiliares que contienen algunos tipos de ficheros) obtenemos:

Tamaño fichero original = 24 bits × 768 × 1024 = 18874368 bits = 2359296 *bytes* = 2304 *Kbytes* (dividimos por 1024).

El factor de compresión será por tanto:

Factor de compresión = FC = 2304 *Kbytes* / 650 *Kbytes* = 35,44

Los bpp pueden calcularse expresando el factor de compresión como cociente entre los bpp de la imagen original y la codificada:

bbp (imagen codificada) = bbp (imagen original) / FC = 24/35,44 = 0,68 bpp

Calidad de la imagen. Error cuadrático medio

Para comparar la calidad entre la imagen original y la imagen comprimida puede utilizarse el error cuadrático medio entre las dos imágenes. El error cuadrático medio se define mediante la siguiente ecuación:

error cuadrático medio = mse (*mean square error*)

$$\varepsilon^2(\text{mse}) = \frac{1}{N \times M} \sum_{n=1}^{N} \sum_{m=1}^{M} (u[n,m] - w[n,m])^2$$

donde u[n,m] representa la imagen original y w[n,m] la imagen comprimida. Generalmente, en la medida del error cuadrático sólo se comparan las componentes de luminancia de ambas imágenes y se supone que la calidad colorimétrica de la imagen será comparable al resultado obtenido con la luminancia.

El error cuadrático medio es tanto más pequeño cuanto menor es la diferencia entre la imagen original y la comprimida. El valor obtenido es el promedio entre todos los píxeles de la imagen. Valores típicos de 4 ó 5 unidades de error cuadrático medio resultan indistinguibles desde un punto de vista visual.

No obstante, debe tenerse en cuenta que el resultado final es sólo un valor promediado entre muchos píxeles, por lo que puede ocurrir que el error en algunas regiones de la imagen sea considerable (visualmente perceptible) y que con el promedio se obtenga un error total pequeño. Por ello, la medida del error cuadrático es sólo un indicativo de la calidad de la imagen y no es posible garantizar que cuando el error es pequeño no podrán apreciarse diferencias visuales entre las dos imágenes.

Calidad de la imagen. PSNR (*Peak Signal to Noise Ratio*)

La PSNR es una medida alternativa de la calidad de la imagen que también está basada en el error cuadrático medio. En este caso, el parámetro de medida se calcula como:

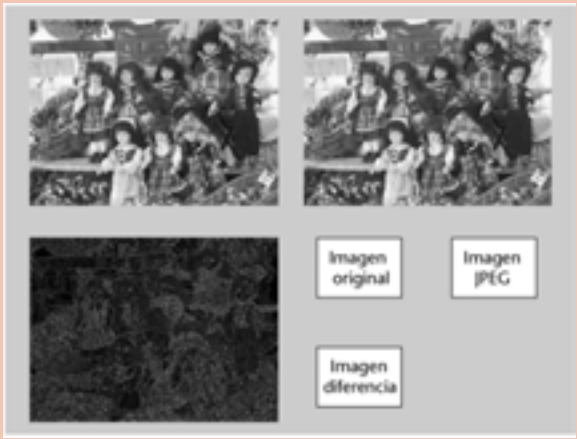
$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{255^2}{\varepsilon^2}$$

La PSNR representa la relación entre la potencia máxima de la señal útil (imagen con excursión máxima) y el ruido o error introducido en el proceso de compresión. La PSNR se mide en unidades de decibelio indicando una mejora de calidad a medida que aumenta la relación. Relaciones superiores a 36 dB (decibelios) resultan en imágenes prácticamente indistinguibles. Entre 28-34 dB, las imágenes presentan buena calidad, aunque pueden percibirse pequeñas diferencias. Para relaciones inferiores a 28 dB, los defectos en la imagen comprimida suelen resultar evidentes. Relaciones inferiores a 25 dB suelen ser inaceptables desde un punto de vista visual.

Ejemplo 1. Medidas de calidad en imágenes comprimidas (calidad buena)

En la figura adjunta se compara una imagen original (bitmap) con la imagen comprimida en JPEG. El fichero de la imagen original es de 770 *Kbytes*, mientras que el fichero comprimido es de 63 *Kbytes*. El factor de compresión es, por tanto, de 12,2 y las dos imágenes son aparentemente indistinguibles.

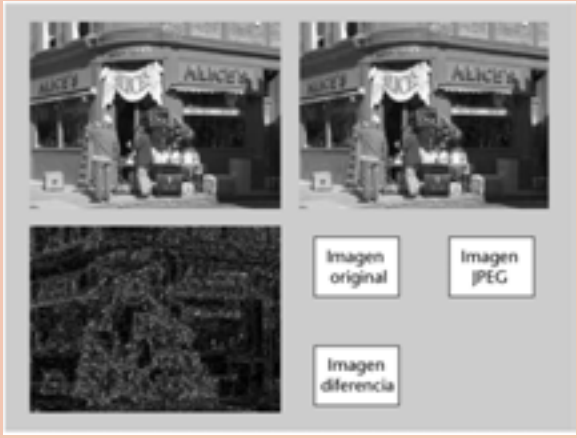
La figura también muestra el error entre las dos imágenes que se concentra principalmente en los detalles y contornos de la imagen original. El error se ha multiplicado por 10 para que resulte visible. El error cuadrático medio entre la imagen original y la comprimida es de 0,45, lo que da una PSNR de 51 dB (calidad muy buena).



Ejemplo 1. Comparación entre imagen original e imagen comprimida con buena calidad

Ejemplo 2. Medidas de calidad en imágenes comprimidas (calidad baja)

En este segundo ejemplo se ha aumentado el factor de compresión y se ha reducido la calidad de la imagen. La imagen original tiene un tamaño de 900 *Kbytes* y se ha comprimido hasta 21 *Kbytes*. El factor de compresión es ahora de 42 y la PSNR resultante de 27 dB. La imagen diferencia sólo ha sido multiplicada por un factor 3, con lo que ya pueden apreciarse las diferencias.



Ejemplo 2. Comparación entre imagen original e imagen comprimida con baja calidad

Características generales del estándar JPEG

El estándar JPEG define varios modos para comprimir la imagen que pueden clasificarse como modos con pérdidas o sin pérdidas. Existe un único modo de compresión sin pérdidas que permite reconstruir la imagen original de forma exacta a partir de la comprimida. Los factores de compresión que se consiguen con este modo son relativamente bajos, aunque algo mejores que los que se conseguirían aplicando algoritmos convencionales de compresión de datos como el WinZip, el pkzip o el arj. El modo de compresión sin pérdidas está basado en la predicción lineal de la imagen y la codificación de la señal diferencia mediante códigos de longitud variable.

En los modos con pérdidas, la imagen original ya no puede recuperarse de forma exacta, pero pueden obtenerse factores de compresión entre 10 y 20 sin pérdidas de calidad aparente. Están definidos 3 modos de compresión con pérdidas cuyos principios de codificación se analizan con detalle en la siguiente etapa:

Modos de compresión con pérdidas	
Modo secuencial base (*Baseline Sequential Mode*)	La imagen se codifica dividiéndola en bloques de pequeño tamaño (8 × 8 píxeles) y siguiendo una secuencia de izquierda a derecha y de arriba abajo. El decodificador recupera la imagen en el mismo orden. Se han definido distintas extensiones de este modo básico.
Modo progresivo (*Progresive Mode*)	La imagen se codifica en múltiples exploraciones utilizando en todos los casos la misma resolución espacial. El decodificador puede obtener de una forma rápida una primera aproximación de la imagen que posteriormente se refina al avanzar el proceso de decodificación. Está pensado para aplicaciones de acceso de datos a centros remotos con líneas de comunicación lentas de forma que el usuario pueda obtener una primera aproximación del contenido de la imagen antes de descargarla completamente.

Modos de compresión con pérdidas	
Modo jerárquico (*Hierarchical Mode*)	La imagen se codifica con distintos niveles de resolución espacial, lo que facilita que pueda ser mostrada en distintos tipos de *display*. La resolución más baja es la primera que se obtiene en el decodificador que posteriormente puede irse aumentando al decodificar el resto de los datos. El usuario puede detener el proceso de decodificación cuando la resolución es la adecuada.

En todos estos modos se utiliza la transformada coseno discreto (DCT) como herramienta básica de compresión que se combina con técnicas de codificación por longitud de series y códigos de longitud variable.

Etapa 2: El estándar JPEG

Introducción

El estándar JPEG está ampliamente difundido y actualmente es el más utilizado para codificar imágenes naturales con diferentes grados de calidad. Sus aplicaciones incluyen, entre otras, las cámaras de fotografía digital, la compresión de archivos fotográficos, la transferencia de imágenes mediante redes informáticas y la captura de vídeo digital (magnetoscopios digitales y algunas tarjetas de captura). El JPEG también está incorporado como codificador y/o decodificador en diferentes productos informáticos como programas para el tratamiento de fotografías, exploradores web, manejadores de archivos fotográficos, tratamiento de textos y gráficos, etc., con lo que se facilita el intercambio de ficheros entre aplicaciones de diferentes desarrolladores.

Sin embargo, como ya hemos comentado, no se trata de un procedimiento de codificación-decodificación único, sino que incluye distintos modos de compresión y extensiones para adaptarse a las necesidades de la aplicación. Actualmente, en las diferentes modificaciones y mejoras del estándar aparecen más de 40 variantes del algoritmo básico. La mayoría de las aplicaciones sólo implementa un conjunto reducido de estos modos de compresión que suele resultar suficiente para el manejo de archivos y el intercambio de datos entre aplicaciones.

En esta etapa describiremos los principios y técnicas de compresión utilizados en los 4 modos más importantes. Dedicaremos una especial atención y detalle al estudio del modo secuencial base, ya que constituye el núcleo principal de todos los modos de compresión con pérdidas.

Modo secuencial base

El modo secuencial base (*baseline mode*) descompone la imagen que hay que codificar en bloques de 8×8 píxeles que se ordenan de izquierda a derecha y de arriba abajo. Para cada uno de los bloques resultantes se determina la transformada coseno discreto (DCT) y el resultado obtenido se recuantifica teniendo en cuenta la sensibilidad del sistema visual humano a las distintas componentes. El usuario puede controlar el factor de compresión o la calidad de la imagen a través de un parámetro que determina la precisión con la que se cuantifican los coeficientes de la transformada coseno. Finalmente, los coeficientes resultantes se exploran y se codifican utilizando códigos de longitud variable distintos para las componentes de continua y alterna.

En la figura siguiente se muestra un diagrama general del proceso de codificación para cada uno de los bloques en los que se descompone la imagen. Veremos los detalles de cada una de las etapas en los apartados siguientes.

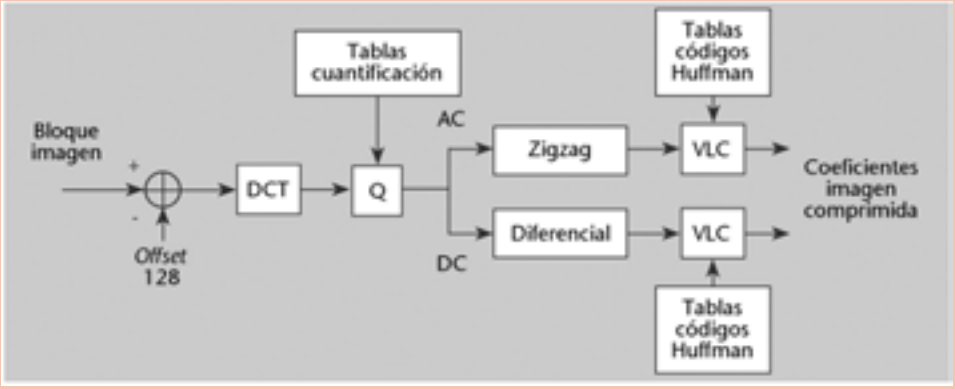


Diagrama del proceso de codificación de un bloque de imagen en JPEG modo base

Descomposición de la imagen en bloques

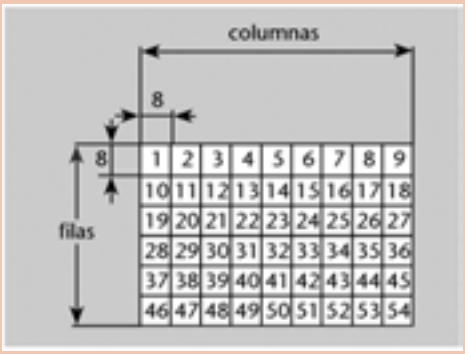
La imagen se divide en bloques de 8×8 píxeles que se codifican de acuerdo con la secuencia indicada en la figura adjunta. La figura representa una imagen de una única componente (luminancia). Para imágenes de color (con varias componentes), los bloques de luminancia y de croma se codifican de forma intercalada. El patrón de intercalación depende del formato de la imagen. Así, para un formato 4:4:4, los bloques correspondientes a Y, Cb y Cr se intercalan directamente. En cambio, si el formato es 4:2:2, existe el doble de bloques de luminancia que de croma (ya que está submuestreada) y la intercalación se realiza codificando 2 bloques de luminancia, uno de Cb y uno de Cr. En un formato 4:2:0 se codificarían 4 bloques de luminancia, uno de Cb y uno de Cr. Se realiza siempre la exploración de todas las componentes de izquierda a derecha y de arriba abajo. En los siguientes apartados supondremos que estamos codificando una imagen con varios niveles de gris. Para codificar imágenes en color debe seguirse el mismo proceso, pero aplicando las operaciones a bloques de componentes de croma en vez de componentes de luminancia.

El tamaño del bloque se selecciona de 8×8 píxeles teniendo en cuenta el compromiso entre capacidad de compresión y complejidad de cálculo. Es conveniente que el tamaño en filas y columnas del bloque sea una potencia de 2 debido a que sólo con estas condiciones existen algoritmos rápidos para el cálculo de la DCT. Por tanto, las posibles alternativas inmediatas al bloque de 8×8 eran el bloque de 4×4 o 16×16.

Un tamaño de bloque mayor proporcionaría una mayor capacidad de compresión de datos ya que la DCT puede extraer mejor la redundancia espacial contenida en cada uno de los bloques. Sin embargo, los cálculos de DCT de mayor orden suponen un aumento considerable de la carga computacional. Uno de los principales objetivos del JPEG era proporcionar técnicas de compresión cuya implementación resultara viable mediante *hardware* de bajo coste o *software* genérico teniendo en cuenta el estado tecnológico de principios de los noventa. Por ello, se decidió utilizar un tamaño de bloque de 8×8 píxeles.

Los bloques están codificados con 8 bits (256 niveles), por lo que tienen un nivel de continua (valor medio) distinto de cero. Para mejorar la codificación de esta compo-

nente de continua se extrae de forma sistemática (se resta) un *offset* de 128 a todos los elementos del bloque antes de calcular la DCT.



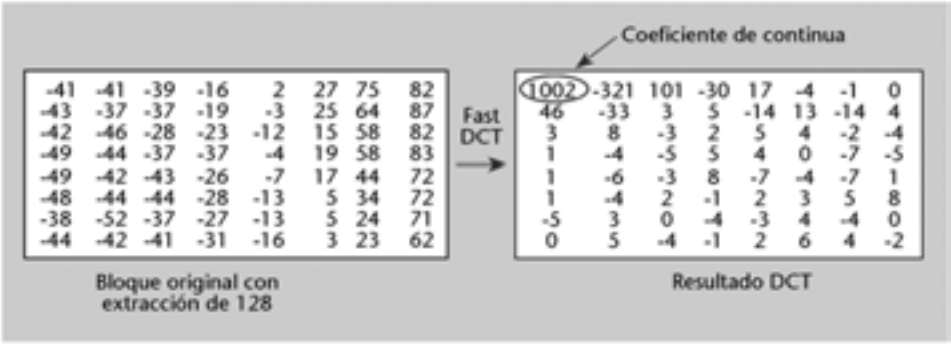
Descomposición de la imagen en bloques de 8 × 8 y orden de codificación de los bloques

Cálculo de la transformada coseno

La transformada coseno se calcula mediante algoritmos rápidos (*Fast Discrete Cosine Transform* - FDCT), con lo cual se obtienen los 64 coeficientes transformados. En la figura se representa el resultado de la transformación de un bloque de imagen indicando la posición de las denominadas componentes de continua y de alterna.

El estándar establece que la precisión en el cálculo de la transformada coseno debe ser de 11 bits, por lo que los coeficientes pueden tomar valores comprendidos entre –1024 y 1023. El coeficiente de continua y los coeficientes de alterna se codifican utilizando procedimientos diferentes.

La razón de esta diferencia es que el coeficiente de continua contiene la información del nivel medio de cada bloque. Es necesario realizar una codificación más precisa para evitar que aparezcan efectos de la división en bloques cuando se realice la reconstrucción de la imagen, ya que resultarían visualmente molestos. Estos efectos de bloque tienen tendencia a aparecer cuando se aumenta mucho el factor de compresión debido a las imprecisiones de la cuantificación, pero aún resultarían mucho más visibles si los coeficientes de continua se codificasen con el mismo procedimiento que los de alterna. La codificación de los coeficientes de alterna es menos crítica y puede simplificarse, con lo que se permite obtener un factor de compresión más elevado.



Cálculo de la transformada coseno sobre un bloque de imagen. El coeficiente de continua se codifica por separado del resto de los coeficientes.

Tablas de cuantificación

Las tablas de cuantificación indican la importancia relativa de cada coeficiente transformado y se utilizan para aproximar el resultado de la transformada coseno. Estas tablas tienen en cuenta la sensibilidad del sistema visual humano a las diferentes componentes frecuenciales. La tabla indica los valores por los que deben dividirse los resultados de la transformada coseno antes de codificarlos. La figura siguiente muestra un posible ejemplo de tabla de cuantificación que puede aplicarse a los coeficientes de luminancia. Esta tabla de cuantificación se denomina *matriz de Lohscheller*.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Ejemplo de tabla de cuantificación para aplicar a los coeficientes transformados de un bloque de luminancia

La tabla se obtiene de forma experimental, a partir de pruebas de calidad subjetivas que tienen en cuenta la respuesta del sistema visual humano. Observad cómo el paso de cuantificación aumenta, es decir, se utilizan menos niveles, a medida que aumentan las frecuencias. Existen diferencias de simetría entre las componentes horizontales y verticales que sólo se justifican a partir de la naturaleza experimental con la que se han obtenido los resultados.

Las matrices o tablas de cuantificación dependen del formato en el que se codifica la imagen y de las componentes de luminancia y color que se codifican. Los valores de la tabla anterior son sólo un ejemplo para bloques de luminancia y, en general, las tablas pueden definirse en el momento de realizar la codificación de la imagen. El estándar permite que la aplicación defina las tablas de cuantificación que se utilizan almacenando los valores en la cabecera del fichero.

Ejemplo de tabla de cuantificación para bloques de croma

Debido a que el sistema visual humano presenta distinta sensibilidad a las componentes de luminancia y de color, las matrices de cuantificación que se utilizan para la codificación de bloques de croma también son distintas. En la figura siguiente se muestra un ejemplo típico de una matriz de cuantificación para componentes de croma.

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Matriz de cuantificación típica para bloques de croma

Uso de las tablas de cuantificación

Desde un punto de vista operativo, las matrices o tablas de cuantificación se utilizan para determinar el nivel real con el que se codificará cada coeficiente transformado. El cálculo del nivel se realiza dividiendo los coeficientes de la DCT por el número de pasos de cuantificación especificados por la matriz de Lohscheller y aproximando el resultado obtenido por el entero más próximo.

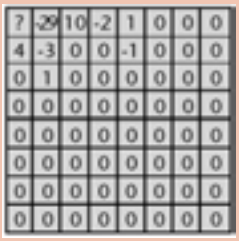
$$V_{uv} = \text{round}\left(\frac{DCT_{uv}}{Q_{uv}}\right)$$

donde V_{uv} representa los coeficientes definitivos que deberemos codificar, DCT_{uv}, los coeficientes originales obtenidos mediante la transformada coseno y Q_{uv}, la matriz de cuantificación.

En la figura adjunta se muestra el resultado de aplicar esta operación al bloque de la DCT que habíamos obtenido anteriormente. Así, el elemento de la primera fila segunda columna se obtiene como resultado de dividir –321 (el valor del coeficiente de la DCT) por 11 (el valor de la tabla de Lhoscheller en la primera fila, segunda columna). Por tanto:

$$V_{12} = \text{round}(-321/11) = \text{round}(-29{,}18) = -29$$

El resto de las componentes se determina de modo análogo, y se obtiene el siguiente resultado final después de realizar todas las divisiones y aproximaciones a enteros. Se omite el valor obtenido para el coeficiente de continua debido a que, como ya hemos mencionado, el procedimiento de codificación de este coeficiente es ligeramente distinto.



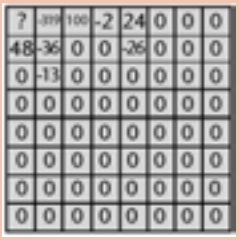
Resultado obtenido al aplicar la matriz de cuantificación de luminancia al ejemplo de la DCT

Observad que la mayor parte de los coeficientes que deben codificarse es nula. Esto significa que podemos combinar estrategias de codificación mediante secuencias de longitud de ceros y códigos de longitud variable de forma muy eficiente.

Uso de las tablas de cuantificación en el decodificador

El lector puede cuestionarse que los valores que finalmente se codifican no tienen una relación directa con los resultados que hemos obtenido al realizar la transformada coseno. En efecto, en realidad los valores que se almacenan en el fichero no son los de la transformada coseno, sino los resultados de dividir estos valores por los valores de la tabla de cuantificación.

No obstante, esta circunstancia no es insalvable puesto que el decodificador dispone de los valores utilizados durante la codificación en la tabla de cuantificación (recordad que los valores están incluidos en la propia cabecera del fichero). De esta manera es posible realizar la operación inversa para recuperar de forma aproximada los valores de la transformada coseno. Si multiplicamos los valores codificados por la matriz de cuantificación, obtenemos:



Valores aproximados obtenidos por el decodificador al realizar la operación de cuantificación inversa

Es importante observar que los valores obtenidos son sólo una aproximación a los valores de la transformada coseno original. Los coeficientes de menor valor absoluto han sido aproximados por cero de modo que la información que contenían ya no puede ser recuperada. Por ello, la imagen que puede obtenerse durante el proceso de decodificación es sólo una versión aproximada de la imagen original.

Los coeficientes obtenidos en el decodificador serán aplicados a una transformada coseno inversa de modo que será posible reproducir, de forma también aproximada, los niveles de gris de la imagen original. Cabe notar también que si un coeficiente de la transformada coseno es suficientemente elevado, será codificado con un valor distinto de cero. A medida que los coeficientes se corresponden con componentes de frecuencias más altas (hacia los extremos inferiores o hacia la derecha), es necesario que tomen valores cada vez más elevados para que sean codificados con valores distintos de cero. Esta circunstancia se debe a la menor sensibilidad del sistema visual humano a las componentes de frecuencia elevada.

Factor de calidad de la compresión JPEG

El factor de calidad o compresión del estándar JPEG puede controlarse directamente actuando sobre el proceso de cuantificación. Para ello se define un factor de calidad *q_JPEG* que se expresa en tanto por ciento y que puede ser introducido directamente por el usuario durante la compresión de una imagen. El factor de calidad puede tomar valores entre el 1% y el 100%.

A partir del factor de calidad especificado por el usuario se calcula una constante α que actúa directamente sobre las tablas de cuantificación. El valor de la constante α se determina a partir del factor de calidad de acuerdo con las siguientes ecuaciones:

$$\alpha = \frac{50}{q\_JPEG} \qquad \text{si } 1 \le q\_JPEG \le 50$$

$$\alpha = 2 - \frac{2 \cdot q\_JPEG}{100} \qquad \text{si } 50 \le q\_JPEG \le 99$$

La constante α toma el valor unidad cuando el factor de calidad es igual al 50%. Para factores de calidad inferiores al 50% la constante α aumenta de forma paulatina y toma un valor igual a 2 para una calidad del 25% y de 50 para una calidad del 1% (límite inferior del parámetro de calidad). Cuando el factor de calidad supera el 50%, la constante α disminuye. Para una calidad del 75% toma el valor de 0,5, que disminuye hasta 0,02 para una calidad del 99%.

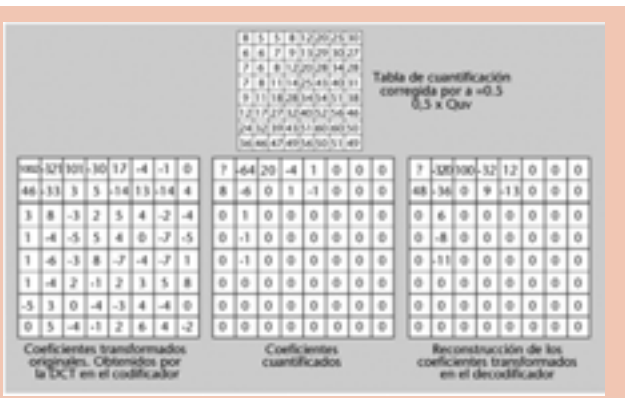
La constante α se utiliza para ajustar, de manera directamente proporcional, los valores de las tablas de cuantificación. Cuando la calidad es del 50% (α es igual a la unidad), las matrices de cuantificación se utilizan tal y como hemos descrito en los apartados anteriores. Sin embargo, cuando la calidad es distinta del 50%, las matrices de cuantificación se corrigen multiplicándolas previamente por el factor α y aproximándolas al entero más próximo. De este modo, cuando el factor de calidad es superior al 50%, los valores de las matrices de cuantificación se corrigen y los coeficientes transformados se dividirán por números más pequeños, con lo que mejora la calidad de la aproximación. Evidentemente, el factor de calidad con el que ha sido codificada una imagen se almacena junto con la cabecera de la imagen de forma que el decodificador siempre puede determinar la tabla de cuantificación exacta que ha sido utilizada durante el proceso de compresión.

Cuando el factor de calidad es exactamente igual al 100%, todos los coeficientes de las matrices de cuantificación toman el valor unidad. Esto significa que para una calidad del 100% no se realizan aproximaciones de los coeficientes transformados y se almacenan (o transmiten) directamente los resultados obtenidos por la DCT.

Ejemplo con factor de calidad del 75%

Para un factor de calidad del 75%, el parámetro α toma un valor igual a 0,5. Esto significa que todos los coeficientes de la matriz de cuantificación tomarán un valor igual a la mitad de su valor nominal. Por ello, al dividir los coeficientes transformados por los nuevos valores de la tabla de cuantificación obtendremos resultados que representan una mejor aproximación a los valores transformados.

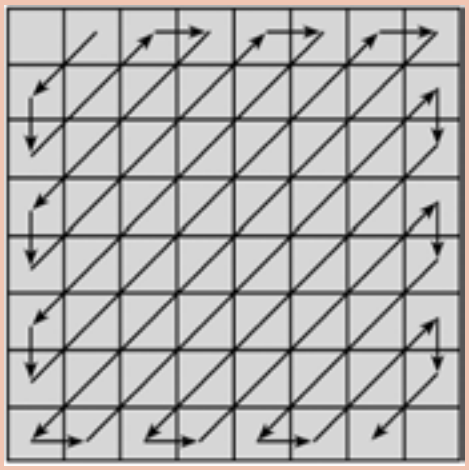
El proceso se ilustra en la siguiente figura donde se proporciona la tabla de cuantificación ponderada por la constante α y los niveles que se obtendrán después de la cuantificación de los coeficientes. La figura también muestra los coeficientes obtenidos después de que el decodificador invierta el proceso de cuantificación. Puede observarse que las aproximaciones a los coeficientes originales son más exactas y que algunos de los coeficientes que anteriormente tomaban un valor nulo ahora son distintos de cero. Esto indica, como era de esperar, que la mejora en calidad representa un aumento del número de coeficientes que deben almacenarse y, por lo tanto, una reducción en el factor de compresión.



Ejemplo de codificación de un bloque de luminancia aumentando el factor de calidad del compresor.

Codificación de los coeficientes de alterna. Exploración en zigzag

Para codificar los coeficientes de alterna se ordenan en zigzag según el esquema de la figura adjunta. Esta ordenación permite que los componentes de baja frecuencia (los más próximos al coeficiente de continua) se almacenen o transmitan en primer lugar. Generalmente estos coeficientes son los más importantes y suelen tomar valores distintos de cero. A medida que aumenta la frecuencia, la probabilidad de que los coeficientes cuantificados tomen valores nulos es más alta, por lo que se producen cadenas de ceros consecutivas que pueden codificarse de forma eficiente indicando el número de ceros que se producen entre dos valores cuantificados.



Secuencia de exploración de los coeficientes de alterna

Ejemplo de codificación de un bloque de coeficientes de alterna

En la siguiente figura se representan los valores de los coeficientes de alterna que deben ser codificados. Suponemos que estos valores son los que se obtienen para un bloque de luminancia después de haber aplicado la tabla de cuantificación de Lhoscheller, con un factor de calidad del 50% y que hemos aproximado los resultados por números enteros. Siguiendo el patrón de exploración en zigzag de los coeficientes obtenemos la siguiente lista de números.

{12, 156, 13, 0, 0, 0, 5, 0, 0, 0, 0, 3, 0, 0, 0, 0, ..., 0 }

Esta secuencia contiene coeficientes que estadísticamente tienen valores próximos a cero. Además también tiene un elevado número de ceros consecutivos. Todo ello indica que podremos combinar técnicas de codificación de longitud de ceros y códigos de longitud variable para reducir el número total de bits con el que se codifica este bloque de información.

?	12	0	0	0	0	0	0
156	0	5	0	0	0	0	0
13	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Ejemplo de la secuencia de codificación de los coeficientes de alterna

Codificación de los coeficientes AC

Para codificar los coeficientes de alterna se utilizan dos símbolos que se asocian a cada uno de los coeficientes distintos de cero. El primer símbolo contiene información sobre el número de ceros que precede a un coeficiente y del tamaño (número de bits) con el que puede codificarse este coeficiente. El segundo símbolo es la codificación en complemento a 1 del valor del coeficiente. El primer símbolo se conoce con el nombre de RUN/SIZE y el segundo como VALUE. Cuando, después de codificar un coeficiente, el resto de los valores es nulo, no es necesario continuar con la codificación del bloque, lo que se indica con un código especial de final de bloque (EOB – *End Of Block*).

Para el ejemplo del apartado anterior, la secuencia de símbolos que deben codificarse es:

{R/S(0,4), VALUE(12), R/S(0,8), VALUE(156), R/S(0,4), VALUE(13), R/S(3,3), VALUE(5), R/S(4,2), VALUE(3), EOB}

Así, el primer código R/S(0,4) indica que al primer coeficiente de AC le preceden 0 números con valor nulo y que el valor del coeficiente se codificará con 4 bits. A continuación se proporciona el valor exacto del coeficiente mediante el código VALUE(12) (el valor binario 1100). El código R/S(3,3) indica que a este coeficiente (con valor numérico 5) le preceden 3 ceros (cuando se realiza la exploración en zigzag) y que será codificado utilizando 3 bits.

Una de las restricciones de esta estrategia de codificación es que el número máximo de ceros consecutivos para el que se admite un símbolo es de 15. Si los coeficientes transformados de un bloque presentan más de 15 ceros consecutivos antes del final de bloque, se utiliza el símbolo especial R/S(15,0) que indica que existen 15 ceros consecutivos que preceden a un número de valor nulo. Así, el símbolo R/S(15,0) se usa para indicar la presencia de 16 ceros consecutivos y puede concatenarse con otros símbolos para codificar secuencias de ceros más largas.

Códigos de longitud variable para símbolos R/S

Para codificar los símbolos R/S se utilizan códigos de longitud variable que han sido optimizados teniendo en cuenta las estadísticas de los datos. El estándar proporciona tablas recomendadas para la codificación de los símbolos R/S. En la figura siguiente se muestra un fragmento de estos códigos para algunos de los símbolos R/S más frecuentes.

A(R/S)	Código Huffman	A(R/S)	Código Huffman
0/0 EOB	1010	1/8	1111111110000110
0/1	00	1/9	1111111110000111
0/2	01	1/10	1111111110001000
0/3	100	2/1	11100
0/4	1011	2/2	11111001
0/5	11010	2/3	1111110100
		2/4	111111110100
0/9	1111111110000010	2/5	1111111110001001
0/10	1111111110000011	2/6	1111111110001010
1/1	1100	2/7	1111111110001011
1/2	11011	2/8	1111111110001100
1/3	1111001	2/9	1111111110001101
1/4	111110110	2/10	1111111110001110

Tablas de Huffman para códigos R/S en coeficientes AC

Codificación del valor entero de los coeficientes de alterna

Para determinar el número de bits con el que se codifica un coeficiente de alterna, es necesario expresar el nivel que hay que codificar en valor absoluto y en binario. El número de bits que se deben codificar es igual al número de bits significativos, es decir, el número de bits que aparecen a la derecha del primer uno incluyendo a éste.

Así, para calcular el número de bits con que debe codificarse el –13 expresamos su valor absoluto (13) en binario:

....0001101

lo que nos indica que debe ser codificado con 4 bits. (El número de bits a la derecha del primer uno significativo incluyéndolo.) Evidentemente, el valor positivo 13 también se debe codificar con 4 bits.

Los códigos asignados a los enteros positivos coinciden con el código binario natural. Así, los bits con los que se codifica el número 13 serán 1101.

Los códigos asignados a los enteros negativos coinciden con el código binario natural de su valor absoluto cambiando todos los 1 por 0 y los 0 por 1 (complemento a 1). Tomando como ejemplo el número –13, su codificación sería 0010.

Para calcular el valor representado por un código binario que representa un coeficiente de alterna, debemos decodificar primero el símbolo R/S que precede a la codificación del número para conocer el número de ceros que le preceden y el número de bits con el que está codificado. El valor del coeficiente es positivo si los bits que lo representan empiezan por 1 y negativo en caso contrario. Cuando el coeficiente es positivo, podemos determinar su valor calculando el valor del código binario na-

tural que lo representa. Cuando es negativo, es necesario cambiar todos los 1 por 0 y los 0 por 1 antes de realizar esta conversión.

Ejemplo. Codificación binaria de los coeficientes de alterna

Tomemos como ejemplo de codificación la secuencia de símbolos que se ha proporcionado anteriormente como ejemplo. La secuencia de los coeficientes explorada en zigzag es:

{12, 156, 13, 0, 0, 0, 5, 0, 0, 0, 0, 3, EOB}

donde ya hemos incluido el código EOB después del último coeficiente no nulo.

El primer coeficiente 12 puede expresarse con 4 bits. Como no existe ningún cero que le precede, le corresponde un símbolo R/S(0,4) y a continuación el VALUE(12). Análogamente, el 156 puede codificarse con 8 bits y no tiene ningún cero que le precede, por lo que le corresponderá el código R/S(0,8) seguido del VALUE(156). Continuando con este procedimiento obtenemos la lista de símbolos R/S y VALUE que ya habíamos puesto anteriormente como ejemplo:

{R/S(0,4), VALUE(12), R/S(0,8), VALUE(156), R/S(0,4), VALUE(13), R/S(3,3), VALUE(5), R/S(4,2), VALUE(3), EOB}

Los códigos binarios para cada uno de estos símbolos pueden extraerse de la tabla de Huffman y del cálculo de las representaciones binarias de los números enteros. Obtenemos:

R/S(0,4):	1011
VALUE(12):	1100
R/S(0,8):	1111 1101 10 (no está en la tabla resumen)
VALUE(156):	1001 1100
R/S(0,4):	1011
VALUE(13):	1101
R/S(3,3):	1111 1111 0101 (tampoco está en la tabla resumen)
VALUE(5)	101
R/S(4,2):	1111 1110 00 (tampoco está en la tabla resumen)
VALUE(3)	11
EOB:	1010

Observad cómo los símbolos R/S que son menos probables desde un punto de vista estadístico tienen asignados códigos de mayor longitud de bits que los símbolos más probables. El código R/S(0,4) sólo tiene asignados 4 bits, mientras que el R/S(3,3) requiere 12 bits.

La secuencia de bits que finalmente se utilizará para codificar todos los coeficientes de alterna del bloque es:

{12, 156, 13, 0, 0, 0, 5, 0, 0, 0, 0, 3, EOB } ==
{1011 1100 1111 1101 10 1001 1100 1011 1101 1111 1111 0101 101 1111 1110 00 11 1010}

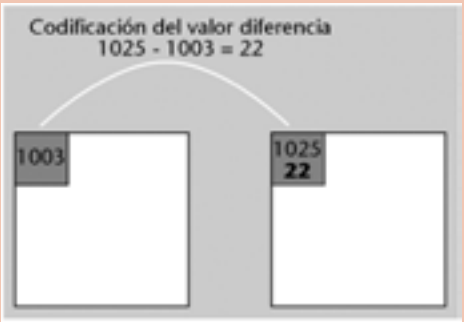
Para comprobar la eficiencia de este esquema de codificación debemos observar que toda la secuencia de 64 coeficientes de este ejemplo ha sido codificada con tan sólo 65 bits. Si hubiéramos codificado cada uno de los coeficientes con los 11 bits originales que proporciona la DCT, se requerirían $11 \times 64 = 704$ bits.

Cabe notar que en la codificación de algunos números se obtiene mayor ganancia que en otros. El primer 12 está codificado con un total de 8 bits 1011 1100 (el R/S más el VALUE), mientras que el 156 requiere un total de 18 bits (más bits de lo que requeriría con una codificación directa). Esto se debe a que el valor de 156 es muy poco habitual como coeficiente transformado, por lo que se producirá estadísticamente muy pocas veces. En cambio, la secuencia de 0 0 0 3 (un total de 4 coeficientes) se codifica tan sólo con 12 coeficientes (un promedio de 3 bits por cada coeficiente).

Los códigos de Huffman asignados a los símbolos R/S y la codificación binaria de los valores de los coeficientes ha sido cuidadosamente estudiada durante la elaboración del estándar para que produzca secuencias de bits que estadísticamente comprimen los datos con gran eficiencia.

Codificación de los componentes de continua (DC)

Los coeficientes de continua se codifican de forma diferencial, tomando la diferencia respecto al coeficiente de continua del bloque anterior. Este proceso se ilustra en la figura adjunta donde se muestra que si el coeficiente de continua en el bloque anterior tomaba el valor 1003 y en el bloque actual toma el valor 1025, el valor que se codificará es la diferencia entre ambos, es decir, 22. En el primer bloque de la imagen se codifica directamente el valor del coeficiente de continua.



Codificación diferencial del valor de continua

La diferencia se codifica de forma parecida a los coeficientes de alterna utilizando la combinación de dos símbolos. El primer símbolo (S-*Size*) indica el número de bits necesario para codificar la diferencia entre los niveles de continua y el segundo codifica en binario natural en complemento a uno este valor (igual que en el caso de los coeficientes de alterna).

Para codificar los símbolos S, que indican el número de bits con el que se codificará la componente de continua, se utilizan tablas de Huffman. En la siguiente gráfica se

muestran los códigos de Huffman recomendados por el estándar para codificar los símbolos S.

Códigos de Huffman para codificar los símbolos S en los coeficientes de continua	
S	**Código**
0	00
1	010
2	011
3	100
4	101
5	110
6	1110
7	11110
8	111110
9	1111110
10	11111110
11	111111110

Cabe observar que cuando la diferencia entre dos coeficientes de continua es cero, no es necesario codificar el valor, sino simplemente proporcionar el código S = 0 (2 bits). Los dos bits 00 indican que el coeficiente de continua del bloque actual es igual al coeficiente de continua del bloque anterior.

Igual que en el caso de las tablas de Huffman de los coeficientes de alterna, el compresor puede determinar las tablas óptimas para cada imagen y almacenarlas en la cabecera del fichero. No obstante, es más habitual utilizar las tablas por defecto que recomienda el estándar ya que han sido analizadas con sumo detalle y presentan excelentes prestaciones para la mayoría de las imágenes.

Ejemplo. Codificación completa de un bloque de coeficientes transformados

En la figura adjunta se muestra el proceso de codificación completo de un bloque de coeficientes transformados. Se supone que los valores que se codifican corresponden al resultado de transformar mediante la DCT un bloque de 8×8 píxeles y aplicar las tablas de cuantificación de Lhoscheller.

El coeficiente de continua que se ilustra en el bloque que hay que codificar (valor 4) es la diferencia entre el coeficiente de continua del bloque actual y el del bloque anterior.

Para codificar este valor debemos tener en cuenta que el número 4 puede representarse con 3 bits. Por ello, el código S deberá ser igual a 3, que corresponde con un código de Huffman de tres bits 100. El valor en binario natural del 4 es 100. Por lo tanto, la secuencia asignada para codificar el nivel de continua es 100 100.

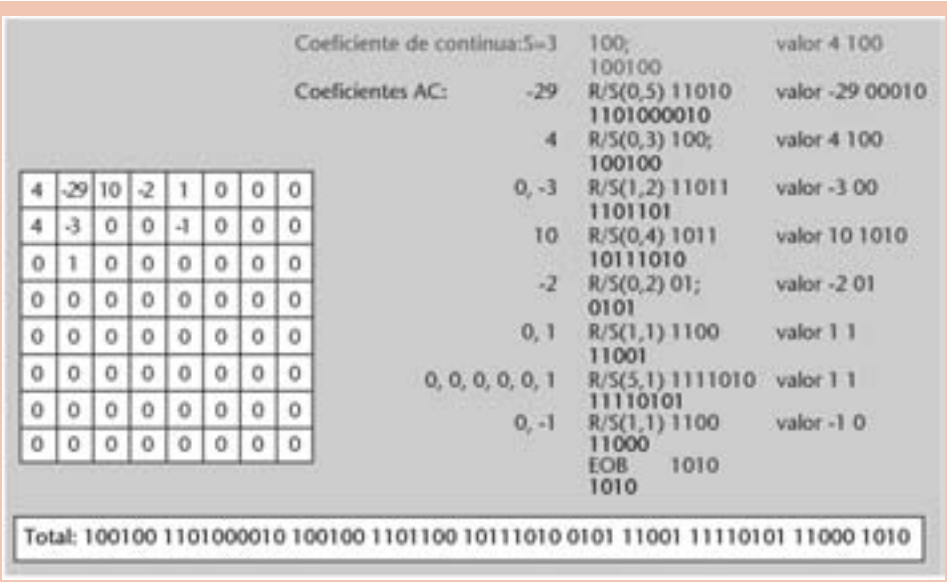
La lista de coeficientes de alterna explorada en zigzag es:

$$\{-29,\ 4,\ 0,\ 3,\ 10,\ -2,\ 0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ -1,\ \text{EOB}\}$$

Si convertimos esta lista a códigos R/S y VALUE, obtenemos:

{R/S(0,5), VALUE(–29), R/S(0,3), VALUE(4), R/S(1,2), VALUE(3), R/S(0,4), VALUE(10), R/S(0,2), VALUE(–2), R/S(1,1) VALUE(1), R/S(5,1), VALUE(1), R/S(1,1), VALUE(–1), EOB}

Los códigos exactos para cada uno de los símbolos se proporcionan en la figura. Cabe notar que el código binario natural de 29 es 11101, por lo que su negativo se codifica cambiando todos los ceros por unos y viceversa: 00010. El resto de los códigos puede obtenerse de las tablas proporcionadas en los subapartados anteriores.



Ejemplo de codificación completa de un bloque de coeficientes transformados

Esquema básico del decodificador

El proceso de decodificación es inverso al de codificación y resulta trivial una vez comprendido el primero. El decodificador debe obtener las tablas de Huffman que han sido utilizadas durante el proceso de codificación. Con estas tablas es posible obtener los valores cuantificados de los componentes transformados en bloques de 8×8 elementos. Para el coeficiente de continua debemos sumar el coeficiente obtenido para el bloque anterior al valor que se deduce de la trama codificada. Una vez obtenidos los coeficientes y conociendo el factor de calidad con el que se ha codificado la imagen (también incluido en la cabecera del fichero) se aplican las tablas de cuantificación de Lhoscheller para obtener la aproximación a los coeficientes reales de la transformada coseno. A estos coeficientes se aplica una transformada coseno inversa y se suma una constante de 128 a todos los píxeles del bloque (el *offset* que inicialmente había se había introducido en el codificador). Los valores obtenidos se aproximan a enteros y se limitan al margen dinámico entre 0 y 255 (es posible que por errores de cálculo o aproximaciones de los coeficientes algunos píxeles desborden el margen dinámico teórico). Finalmente se recompone la imagen final a partir de los diferentes bloques.

En la figura siguiente se esquematiza el proceso de decodificación del modo secuencial base del JPEG.

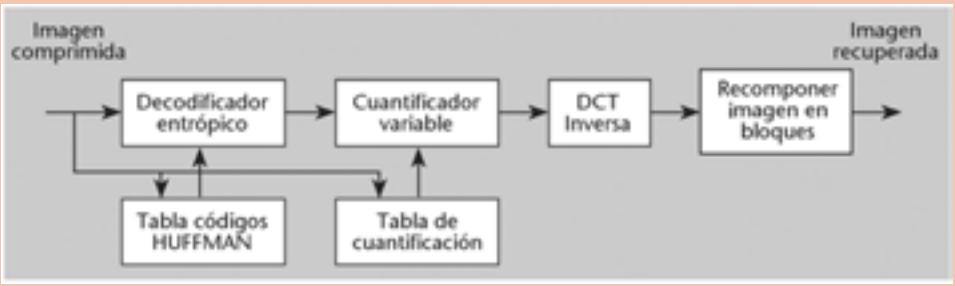


Diagrama genérico de un decodificador JPEG en el modo secuencial base

Resumen de características del modo secuencial base

El modo secuencial base soporta distintas técnicas de compresión elementales que suelen ser suficientes para un elevado número de aplicaciones. No obstante, el estándar define más de 40 posibles extensiones de este procedimiento para optimizar la relación entre el factor de compresión y la calidad de imagen o adaptarse a las características de determinadas aplicaciones. Los aspectos más significativos del modo secuencial base son:

- El proceso de compresión está basado en la transformada coseno discreto (DCT).
- La imagen se subdivide en bloques de 8×8 píxeles.
- Cada una de las componentes de la imagen (luminancia, color) están codificadas con 8 bits.
- Los bloques transformados se transmiten o almacenan de forma secuencial. Esto significa que antes de codificar los coeficientes de un bloque de la imagen deben haberse codificado todos los coeficientes del bloque anterior.
- Se utiliza una codificación de la longitud de las secuencias de ceros y el tamaño en bits de los coeficientes basada en códigos de Huffman. La aplicación puede configurar sus propias tablas de Huffman. Pueden definirse dos tablas de Huffman (una para luminancia y otra para componentes de croma) para codificar los tamaños (S) de los coeficientes de continua y 2 tablas para codificar los R/S de los coeficientes de alterna (una para luminancia y otra para croma). El estándar proporciona tablas de ejemplo que en general proporcionan excelentes resultados y que se utilizan a menudo en la mayoría de las aplicaciones.
- Las diferentes componentes de la imagen (luminancia y croma) pueden codificarse de forma entrelazada o no entrelazada. Una codificación no entrelazada significa que todos los bloques de una componente se codifican de forma consecutiva y, a continuación, se codifican los bloques de la otra componente. En la explora-

ción entrelazada se intercalan los bloques de cada componente. El patrón de intercalado depende del formato en que esté codificada la imagen. Así, en un formato 4:4:4 para cada bloque se luminancia se codifica uno de croma Cb y uno de croma Cr. En cambio, en el formato 4:2:0 se codifican 4 bloques de luminancia consecutivos y posteriormente uno de Cb y otro de Cr. La razón de este patrón de intercalado radica en que el número de bloques de una de las componentes puede ser superior al de las otras.

- Los decodificadores deben ser capaces de procesar imágenes codificadas con un máximo de 4 componentes (p.ej., Y, Cr, Cb y transparencia).

Modos extendidos

El estándar JPEG define diferentes modos extendidos para proporcionar características específicas que resultan necesarias en algunas aplicaciones. Generalmente, el modo secuencial base resulta suficiente para la mayor parte de los problemas y es el modo que se incorpora en casi todos los programas de tratamiento fotográfico.

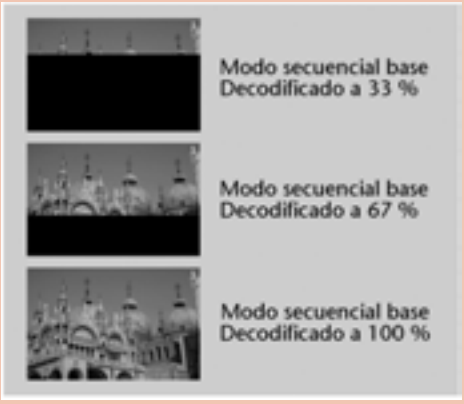
Uno de los modos extendidos permite trabajar con imágenes en las que las componentes originales están codificadas con 12 bits. Este modo es especialmente útil cuando se trata de comprimir imágenes originales que requieren realizar presentaciones de gran calidad.

Los modos extendidos también pueden mejorar la capacidad de compresión de la imagen manteniendo la calidad. Para ello permiten utilizar codificación aritmética en sustitución de los códigos de Huffman en el proceso de codificar las secuencias de coeficientes. La codificación aritmética, como la de Huffman, está basada en un análisis de la entropía de la fuente y aunque es ligeramente más compleja que los códigos de Huffman, permite obtener unos factores de compresión superiores. También es posible definir un total de 4 tablas de codificación para los coeficientes de continua y 4 tablas para los coeficientes de alterna. Las tablas pueden contener códigos de Huffman o tablas aritméticas. Cada una de las tablas puede utilizarse para codificar una componente de la imagen (se permite codificar hasta un total de 4 componentes).

Modo progresivo

Una posibilidad de los modos extendidos es la codificación progresiva que permite transmitir una primera aproximación de la imagen con pocos bits y posteriormente ir mejorando la calidad. El modo progresivo es especialmente útil en aplicaciones de acceso a bases de datos de imágenes fotográficas de alta calidad ya que permite obtener una primera aproximación al contenido global de la imagen antes de obtenerla completamente. Esta posibilidad tiene un gran interés durante el proceso de selección de imágenes debido a que las imágenes pueden ser descartadas mientras aún se están descargando. También es útil en páginas web de acceso lento.

La idea básica del modo progresivo se ilustra en las siguientes figuras en las que se compara la decodificación de una imagen utilizando el modo secuencial base con la de una imagen progresiva. En el primer caso, el decodificador puede mostrar en pantalla los bloques en el orden en que se están decodificando. Si la transmisión es lenta, la imagen va apareciendo paulatinamente de izquierda a derecha y de arriba abajo y sólo disponemos de una perspectiva global de la misma una vez finalizado el proceso de descarga y decodificación. En las figuras se ilustra la decodificación de una imagen de este tipo cuando se ha transmitido un 33% de los datos, un 66% y un 100%. En cambio, cuando se realiza una codificación progresiva, la calidad de la imagen mejora a medida que se van recibiendo nuevos datos presentando desde las primeras fases de la descarga una perspectiva global.



Secuencia de descarga de un archivo en el modo secuencial base

Descarga de un archivo en el modo progresivo

Estrategias para la codificación progresiva

Uno de los principales problemas de la codificación progresiva es que la información asociada a la imagen se transmite de forma escalonada, y no se dispone de toda la información completa asociada a un bloque hasta prácticamente el final de la trans-

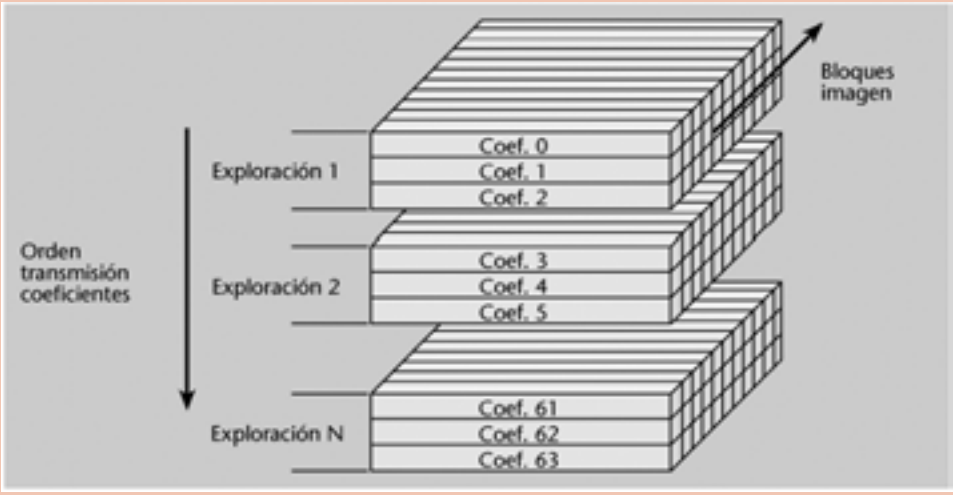
misión. Esta circunstancia exige que tanto el codificador como el decodificador deban disponer de un *buffer* de memoria considerable en el modo secuencial base, ya que la información enviada en las primeras fases de la transmisión debe almacenarse temporalmente hasta que se reciba el resto. Existen dos estrategias para la codificación progresiva de la imagen que dependen de cómo se organizan los coeficientes y en qué orden se transmiten los bits.

- Codificación espectral
- Codificación por aproximaciones sucesivas

Algunos esquemas de codificación progresiva combinan ambas estrategias.

Codificación espectral

En la codificación espectral se realizan distintas exploraciones de los bloques transformados de la imagen en el orden convencional. No obstante, en cada una de las exploraciones se envían únicamente unos pocos coeficientes. En la siguiente figura se muestra una posible estructura de codificación. En este ejemplo, en la primera exploración de la imagen se envía el coeficiente de continua y los dos primeros coeficientes de alterna (en el orden de exploración de zigzag). A partir de estos tres coeficientes, el decodificador puede obtener una primera aproximación de baja frecuencia de la imagen. Posteriormente, en la siguiente exploración se envían los coeficientes de alterna 3, 4 y 5. Cuando el decodificador recibe estos coeficientes, puede recomponer la imagen con todos los coeficientes recibidos hasta este momento mejorando la calidad progresivamente hasta que se han transmitido todos los coeficientes significativos.



Estructura de codificación de la imagen en una codificación espectral progresiva

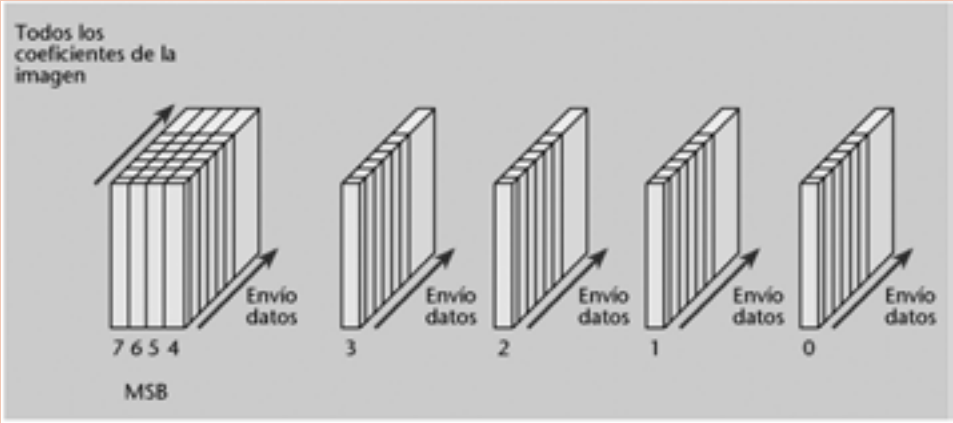
El orden de codificación que se ha utilizado en este ejemplo es simplemente orientativo, ya que el codificador puede seleccionar la estructura de codificación progresiva que considere más adecuada. Una alternativa es transmitir en la primera exploración únicamente el coeficiente de continua, en la segunda los dos primeros

coeficientes de alterna, en la siguiente fase los 4 coeficientes de alterna siguientes, en la siguiente 8 coeficientes, etc. Esta estructura de codificación se denomina *espectral*, ya que la imagen se transmite utilizando bandas de frecuencia independientes en cada una de las exploraciones. En la primera exploración se envían las componentes de frecuencia más baja, posteriormente la banda de frecuencias bajas-medias, etc.

Codificación por aproximaciones sucesivas

En este caso la estrategia de codificación se basa en realizar distintas exploraciones de la imagen transmitiendo en primer lugar los bits más significativos de los coeficientes transformados. A medida que se realizan nuevas exploraciones de la imagen se proporciona progresivamente el resto de los bits de cada uno de los coeficientes.

En la siguiente figura se ilustra a modo de ejemplo una posible estructura de transmisión. En la primera exploración se transmiten los 4 bits más significativos de todos los coeficientes transformados, mientras que en las exploraciones sucesivas se va enviando el resto de los bits. Evidentemente, el codificador dispone de flexibilidad para elegir otras secuencias de exploración. Un ejemplo alternativo sería que en la primera exploración sólo se transmitiera el bit más significativo de todos los coeficientes, en una segunda exploración el segundo bit, etc. Generalmente la exploración mediante aproximaciones sucesivas proporciona una mejor calidad en las primeras fases de la decodificación que la codificación espectral, aunque naturalmente, requiere un mayor volumen de datos para transmitir en las primeras fases.



Estructura de codificación progresiva mediante aproximaciones sucesivas

El estándar también permite combinar los dos modos de exploración progresiva. En este caso puede realizarse una codificación espectral en la que los bits asociados a los coeficientes se transmiten también progresivamente. Un posible ejemplo sería enviar primero los 2-3 bits más significativos de la componente de continua, y posteriormente, en la siguiente exploración el resto de los bits de esta misma componente. En la siguiente fase de exploración se enviarían los bits más significativos de los primeros coeficientes de alterna y así sucesivamente.

Modo de codificación sin pérdidas

El estándar JPEG define un modo de codificación sin pérdidas que permite recuperar la imagen original de forma exacta. El modo de codificación sin pérdidas está basado en la codificación diferencial y utiliza predictores de la imagen que permiten reducir la entropía de la fuente. En este modo de codificación no se utiliza la transformada coseno.

El esquema general de codificación se muestra en la figura adjunta. La imagen puede tener cualquier resolución y las componentes se codifican de forma independiente. Puede estar en el formato RGB o en cualquiera de los formatos YCbCr. La precisión de cada componente puede estar comprendida entre 2 y 16 bits.

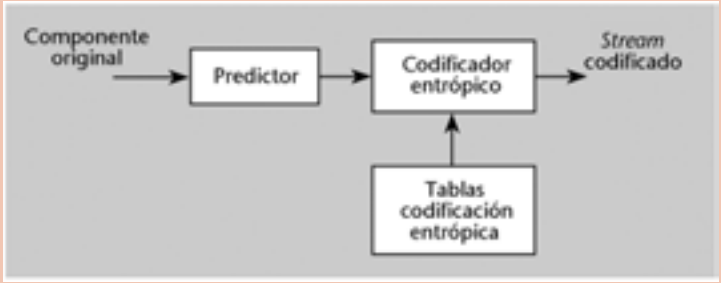


Diagrama general del codificador sin pérdidas

El predictor realiza una estimación del valor de un píxel basándose en los valores de los píxeles situados en su proximidad. En la figura se representa una posible estructura de predicción en la que se combinan los niveles de los píxeles situados en la posición anterior dentro de la misma fila y la fila anterior. Cada uno de los niveles se ponderan con unos coeficientes a, b y c que han sido determinados de antemano y que el codificador indicará en la cabecera del fichero. La señal que realmente se transmite es la diferencia entre la predicción y el valor real del píxel en la posición x. Como el decodificador dispone de los coeficientes utilizados para realizar la predicción, puede aproximar valor del píxel mediante la predicción y utilizar el error transmitido para recuperar el valor original de forma exacta. La expresión de la señal de error viene dada por:

$$e[n, m] = x[n, m] \text{ ñ } (a \cdot x[n, m \text{ ñ } 1] + b \cdot x[n \text{ ñ } 1, m \text{ ñ } 1] + c \cdot x[n \text{ ñ } 1, m])$$

La ventaja de transmitir el error entre el valor real del píxel y la estimación es que la señal resultante tiene menos entropía (estadísticamente todos los errores tienen valores muy pequeños) por lo que pueden utilizarse codificadores entrópicos para reducir la tasa total de bits que es necesario almacenar o transmitir.

Posible estructura de predicción del píxel

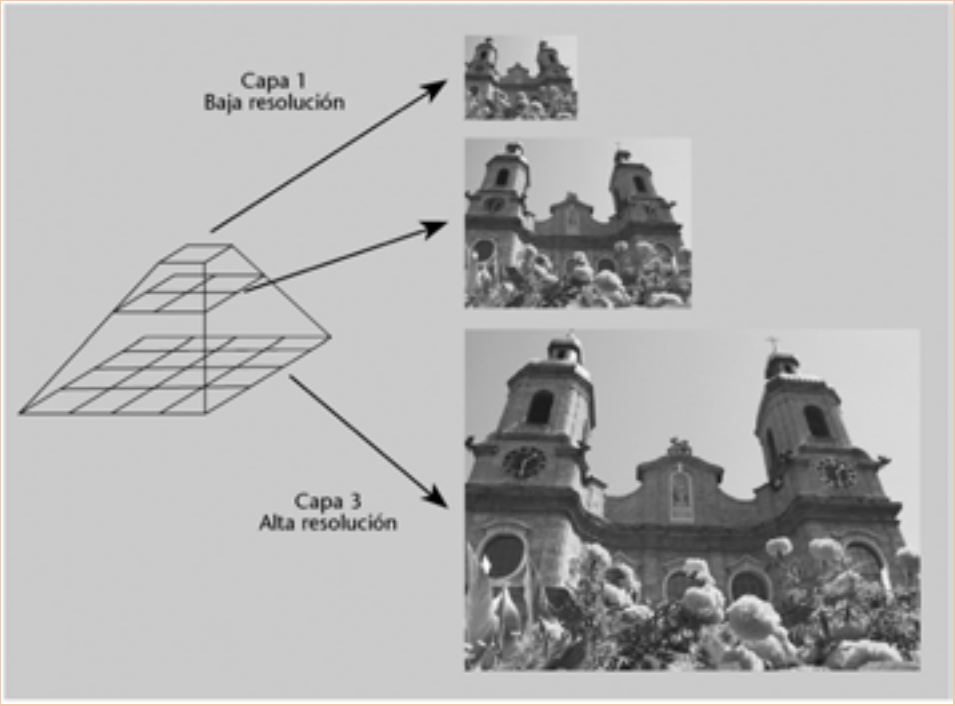
El codificador entrópico puede estar basado en tablas de Huffman que tienen en cuenta la estadística de los errores de predicción. También es posible codificar los errores utilizando codificadores aritméticos (más complejos, pero más eficientes desde el punto de vista de compresión). En general, el factor de compresión obtenido con el método de codificación sin pérdidas es muy inferior al que se obtiene con los métodos con pérdidas basados en la transformada coseno. No obstante, en algunas aplicaciones como las imágenes médicas o algunos archivos de imágenes es necesario almacenar o transmitir la información sin que se produzcan pérdidas. El modo sin pérdidas permite obtener unos factores de compresión algo superiores a los que se obtendrían utilizando compresores de archivos convencionales como el pkzip, winzip, etc., ya que están específicamente adaptados a las características de la imagen.

Modo jerárquico

El modo jerárquico es uno de los más complejos que contempla el estándar. Está pensado para aplicaciones en las que pueda resultar de interés obtener la imagen original con diferentes resoluciones que dependerán del dispositivo final de representación.

La información se estructura en forma de capas que proporcionan información independiente sobre la imagen. La capa base (primera capa) es necesaria para obtener la información del resto de las capas. La resolución de la imagen proporcionada en la capa base es la más baja. Generalmente, la capa base es compatible con el modo secuencial base.

En la figura se ilustra un ejemplo de codificación en el modo jerárquico con 3 capas. La resolución proporcionada por la primera capa permite obtener una imagen de baja resolución. La segunda capa proporciona información adicional para que el decodificador pueda obtener una imagen de mayor resolución. Finalmente, la tercera capa proporciona toda la información para recomponer la imagen a resolución completa. El ejemplo sólo muestra tres capas, pero el estándar es flexible respecto al número total de capas.



Compresión de imagen en el modo jerárquico

El modo jerárquico es parecido al modo progresivo exceptuando que en vez de escalar la imagen en calidad se escala en tamaño. Tiene aplicación principalmente cuando las imágenes deben ser accesibles desde distintos medios. Así, cuando las imágenes deben ser presentadas en dispositivos móviles, la capa base puede proporcionar toda la información necesaria para la reproducción en este tipo de pantallas, sin necesidad de tener que transmitir toda la información asociada a la imagen. En cambio, cuando se presentan en pantallas de alta resolución, puede ser necesario decodificar todas las capas.

La estructura de codificación es relativamente compleja. La imagen original con resolución máxima se debe filtrar paso bajo y reducir al tamaño equivalente que se proporcionará en cada una de las capas. Generalmente, en cada capa se reduce la resolución de filas y columnas en un factor 2 (una cuarta parte de la información). La imagen con la resolución más baja se codifica utilizando, generalmente, el método secuencial base. La trama de bits asociada a la capa más baja puede ser decodificada sin necesidad de enviar el resto de las capas.

La capa siguiente proporciona únicamente la información que no está contenida en la capa base. Para ello, el codificador interpola la imagen obtenida a través de la capa base mediante unos filtros que se especifican en la cabecera. La imagen interpolada se resta con la imagen original a resolución intermedia y el error resultante se codifica para construir esta segunda capa. El decodificador deberá realizar la interpolación (aumentar el tamaño de la imagen obtenida en la capa base) utilizando los filtros especificados en la cabecera del fichero. Una vez obtenida la interpolación se suma la señal codificada en la segunda capa (el error cometido por el decodificador al estimar la imagen de alta resolución) para obtener la imagen real.

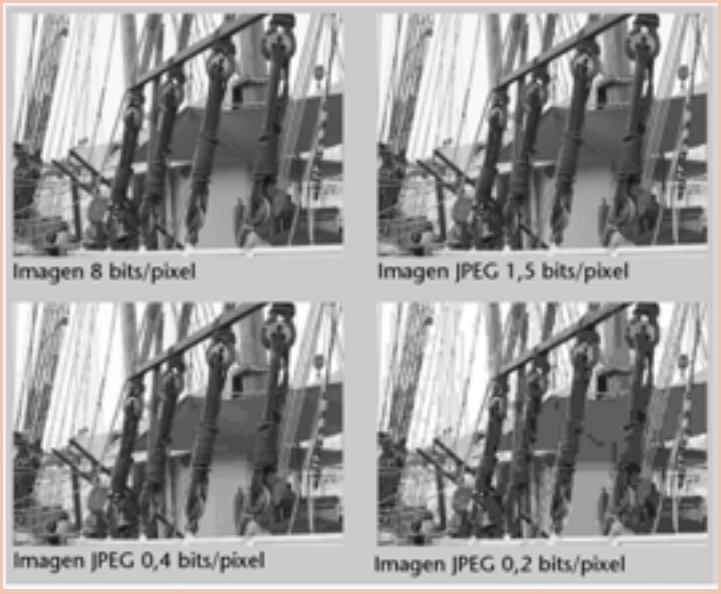
Para codificar las capas altas puede utilizarse la transformada coseno sobre la señal diferencia o un codificador de Huffman o aritmético aplicado directamente sobre la señal de error. En el primer caso, la imagen obtenida una vez decodificada la capa es con pérdidas. En el segundo caso puede obtenerse la imagen sin pérdidas. Generalmente, la codificación de la última capa siempre se realiza sin pérdidas, de modo que pueda obtenerse la imagen original de forma exacta una vez decodificadas todas las capas.

Relaciones de calidad subjetiva obtenidas por el JPEG

El modo básico del JPEG obtiene unos excelentes compromisos entre factor de compresión y calidad de imagen. Las pruebas de calidad subjetiva suelen realizarse con usuarios especialistas que deben valorar entre 0 y 5 la calidad de la imagen reconstruida. Las valoraciones numéricas se corresponden con las siguientes descripciones:

Pobre:	0-1
Moderada:	1-2
Buena:	2-3
Muy buena:	3-4
Excelente:	4
Indistinguible:	5

Entre las calidades *muy buena* y *buena* suelen aparecer efectos de pérdidas en detalles de alta frecuencia. En la calidad *moderada* empieza a apreciarse ligeramente el efecto de bloque y las pérdidas en altas frecuencias son evidentes. En la calidad *pobre*, los efectos de bloque son muy notables.



Comparación de imágenes en blanco y negro codificadas mediante el método base

Relación entre la calidad subjetiva y los bpp

Hemos visto que una imagen en color que utilice las tres componentes R, G y B con una resolución de 8 bits por componente tiene un total de 24 bpp. El formato más habitual de las imágenes sin comprimir es el formato 4:2:2, en el que la luminancia tiene el doble de muestras que cada una de las señales diferencia de color. Estas imágenes están expresadas con 16 bpp. Las imágenes en el formato 4:2:0 tienen 12 bpp. Las calidades subjetivas obtenidas con el JPEG en función de los bpp son:

0,25-0,5 bpp	Moderada-buena
0,5-0,75 bpp	Buena-muy buena
0,75-1,5 bpp	Excelente
1,5-2 bpp	Indistinguibles

En la figura siguiente se muestran algunos ejemplos de imágenes codificadas con JPEG a distintas tasas de compresión. Como puede observarse, los defectos son evidentes en las dos imágenes con mayor factor de compresión.

Codificación de vídeo y el estándar JPEG

El JPEG fue el primer estándar de compresión de imagen que se utilizó en una amplia gama de aplicaciones informáticas y productos *hardware*. Su aplicación principal está en la codificación de imágenes multinivel fijas, aunque también se ha utilizado y se utiliza para la compresión de vídeo. De hecho, el formato JPEG puede considerarse como el precursor de formatos para la codificación de vídeo como el H.261, el MPEG-1, el MPEG-2, el H.263 y el M-PEG4. Las técnicas utilizadas para codificar las imágenes de referencia sin utilizar redundancia temporal en estos formatos tienen las mismas características que las utilizadas en el JPEG.

Muchos formatos de registro de vídeo en cinta magnética profesionales y domésticos utilizan algoritmos basados en el JPEG para comprimir cada uno de los fotogramas de la secuencia de vídeo. Entre estos formatos destacan el DCV-Pro, el DV, Betacam Digital, Mini-Dv, etc.

Cuando se utiliza el algoritmo JPEG para comprimir las imágenes dentro de una secuencia de vídeo, el sistema recibe el nombre genérico de Motion-JPEG (MJPEG). Durante la realización del estándar oficial en la ITU y la ISO, no se consideró ni se documentó ninguna versión del uso del JPEG para la codificación de vídeo fotograma a fotograma, por lo que todos los sistemas existentes en la actualidad son incompatibles y las diferentes versiones del MJPEG se consideran propietarias.

La facilidad para implementar el método secuencial base mediante *hardware* específico de bajo coste ha popularizado las tarjetas de digitalización y compresión de vídeo para PC. Estas tarjetas incorporan un circuito integrado que se encarga de realizar la

compresión de cada uno de los fotogramas de la secuencia de vídeo, registrando el material en el disco duro en formato comprimido. Las tarjetas más conocidas dentro de gama son las DC-30 de Miro, las AVMaster hasta la gama Digisuite de Matrox.

La principal ventaja del formato MJPEG es que todos los fotogramas son codificados de forma independiente, por lo que es posible garantizar la calidad de cada uno de ellos de forma individual. Los formatos de compresión que utilizan compresión temporal (es decir, utilizan otras imágenes de la secuencia para realizar la compresión) pueden proporcionar algunos fotogramas con baja calidad y dificultan la edición y posproducción del vídeo. Por ello, el formato MJPEG se suele seguir utilizando tanto en magnetoscopios profesionales como en tarjetas de digitalización y servidores de vídeo en disco duro, aunque cada vez más se está sustituyendo por versiones específicas del MPEG.

Una de las versiones del MPEG especialmente pensadas para la edición y posproducción de vídeo está basada en utilizar únicamente imágenes del tipo I (Intra) (MPEG-Editable). Estas imágenes forman parte del estándar MPEG y para su codificación sólo se utiliza la información de vídeo existente en el *frame* actual. La codificación de las imágenes I en el formato MPEG es, salvo pequeños detalles en los códigos de longitud variable y en la confección de la trama, idéntica al método secuencial base definido en el JPEG. Actualmente, varios servidores de vídeo para PC utilizan formatos basados en perfiles especiales (*profile* 4:2:2) del MPEG para registrar vídeo con posibilidad de edición profesional combinando el uso de técnicas de compresión de redundancia espacial (como el JPEG) y redundancia temporal.

Etapa 3: El estándar JPEG2000

Introducción

La proliferación de nuevos conceptos multimedia durante la década de los noventa ha motivado el desarrollo de un estándar de codificación de imágenes fijas que incorpora nuevas características que no estaban contempladas en el JPEG y que proporciona una mayor calidad de imagen sobre todo a tasas de codificación muy bajas. El nuevo estándar se denomina JPEG2000 y está basado en la transformada wavelet. Además de permitir una capacidad de compresión mayor que la del estándar JPEG, proporciona un conjunto de funcionalidades adicionales para las que no estaba definido ningún método de compresión. El estándar JPEG2000 pretende cubrir en un único formato de codificación de datos un amplio abanico de aplicaciones que van desde la transmisión de archivos en Internet, la transferencia de fax en color, la fotografía digital, la exploración de imágenes, la digitalización y gestión de imágenes médicas, los archivos fotográficos, la captura remota de imágenes, el comercio electrónico, etc.

El nuevo estándar está pensado para complementarse con el JPEG en aquellas aplicaciones en las que este último presenta pobres resultados. Es probable que poco a poco el JPEG2000 vaya sustituyendo al JPEG hasta desplazarlo, al menos, de forma parcial. No obstante, la introducción del nuevo estándar se prevé que será lenta y que se concentrará principalmente en aplicaciones o conceptos nuevos en los que el JPEG tiene bajas prestaciones. Debemos tener en cuenta que el JPEG está fuertemente implantado y que los resultados que obtiene son más que aceptables en un elevado número de aplicaciones. Aunque en un futuro el estándar JPEG2000 llegara a sustituir completamente al JPEG, este último seguiría teniendo una gran importancia conceptual debido a que ha constituido la raíz sobre la que se han fundamentado otros métodos de compresión de imagen y vídeo como los diferentes estándares MPEG. Actualmente se comercializan distintos compresores–descompresores de JPEG2000 (JP2) que pueden incorporarse como *plugins* en aplicaciones de tratamiento fotográfico, animación gráfica o navegadores web.

La tecnología de compresión utilizada en el JPEG2000 es considerablemente más compleja que la del JPEG. Esta tecnología está basada en la transformada wavelet e incorpora diferentes estrategias para la cuantificación y codificación escalables. Los métodos escalables permiten que el decodificador pueda acceder a la información sin necesidad de interpretar completamente toda la trama de bits que codifica la señal. En esta etapa analizaremos con detalle las distintas prestaciones que incorpora el estándar JPEG2000 desde el punto de vista del usuario. También introduciremos los principios de la transformada wavelet como técnica general para la compresión de información y veremos la arquitectura general del codificador JPEG2000. No obstan-

te, omitiremos muchos detalles del proceso de codificación exacto, ya que requiere unos conocimientos profundos sobre técnicas de tratamiento de señal avanzadas y técnicas de codificación que escapan de los propósitos de este texto.

Características funcionales del JPEG2000

En este apartado detallaremos las principales características funcionales que proporciona el estándar JPEG2000. El objetivo principal es obtener todas estas prestaciones en un único formato de transmisión compatible, sustituyendo los diferentes modos y variantes (hasta 44 variantes distintas) contempladas en el JPEG. Estas prestaciones pueden resumirse en los siguientes puntos:

- Mejora de prestaciones en tasas de transmisión muy bajas.
- Compresión con pérdidas y sin pérdidas combinadas en el proceso de decodificación progresiva.
- Escalabilidad espacial y en SNR.
- Codificación de regiones de interés.
- Acceso aleatorio a una parte de la imagen.
- Arquitectura abierta.
- Combinación de imágenes bi-tonales con imágenes multinivel.
- Robustez frente a errores de transmisión.
- Descripción de alto nivel basada en contenido.

En los siguientes subapartados definiremos los principales objetivos de cada una de estas características.

Mejora en tasas de transmisión muy bajas

El JPEG ofrece resultados con calidades de imagen muy bajas cuando la compresión está por debajo de los 0,25 bpp. El JPEG2000 permite mejorar la calidad de imagen por debajo de este límite, con lo que se obtienen imágenes aceptables (aunque con *artifacts* visibles) hasta 0,1 bpp para niveles de gris. Una de las ventajas del nuevo codificador es que la relación entre la calidad y el factor de compresión se mantiene comparable o incluso superior al JPEG cuando se trabaja con factores de compresión más razonables (aplicaciones en las que la calidad es prioritaria). Las principales aplicaciones que requieren trabajar con tasas de compresión por debajo de los 0,25 bpp son la transmisión de imágenes a través de redes de datos de baja velocidad y las aplicaciones de captura y transmisión de imágenes remotas (*remote sensing*).

En la figura siguiente se muestra el resultado de codificar una imagen mediante la transformada wavelet a una tasa de compresión de 0,1 bpp.

Comparación entre el JPEG y el JPEG2000 en tasas de transmisión muy bajas

Compresión con pérdidas y sin pérdidas combinadas

El estándar permite obtener de forma progresiva versiones con pérdidas de una imagen, cada vez con mayor detalle a medida que avanza el proceso de decodificación. Eventualmente, en la última fase de la decodificación la imagen obtenida puede ser sin pérdidas. Esta característica es parecida a la que se podía obtener utilizando el modo jerárquico de JPEG. La ventaja en este caso es que cualquier decodificador compatible con JPEG2000 debe interpretar la información correctamente (para el JPEG se necesitaba disponer de un decodificador compatible con el modo jerárquico). Otra ventaja es que las primeras etapas de la decodificación requieren un menor volumen de datos para obtener una buena aproximación final a la imagen. En efecto, la mayor capacidad del JPEG2000 para trabajar con tasas de transmisión muy bajas permite obtener buenas imágenes con un número de bits considerablemente menor que el JPEG.

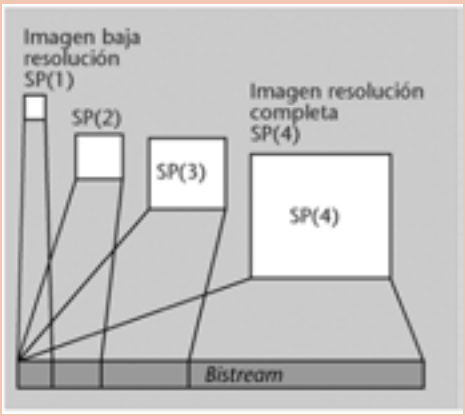
Las aplicaciones que requieren este tipo de codificación son principalmente las imágenes médicas y científicas, que aunque en las primeras fases se puede trabajar con imágenes con pérdidas, el diagnóstico final siempre es conveniente realizarlo con una versión sin pérdidas. Otra aplicación son los archivos de bases de datos fotográficas. A menudo resulta suficiente con visualizar imágenes con pérdidas, pero es importante garantizar que existe una versión de la imagen, accesible cuando se desee, sin pérdidas. Las versiones con pérdidas y sin pérdidas están contenidas en un mismo *bitstream* que puede decodificarse de forma parcial.

La figura siguiente muestra una decodificación progresiva (en la primera etapa y la última) de un archivo de imagen. La primera etapa está codificada con 0,1 bpp de forma que la imagen puede obtenerse de forma muy rápida. La última etapa es la imagen sin pérdidas. Cabe notar que en la imagen de 0,1 bpp los *artifacts* son considerables. No obstante, con esta tasa de compresión en una imagen en color el JPEG convencional no puede trabajar.

Ejemplo de codificación progresiva combinada con pérdidas y sin pérdidas

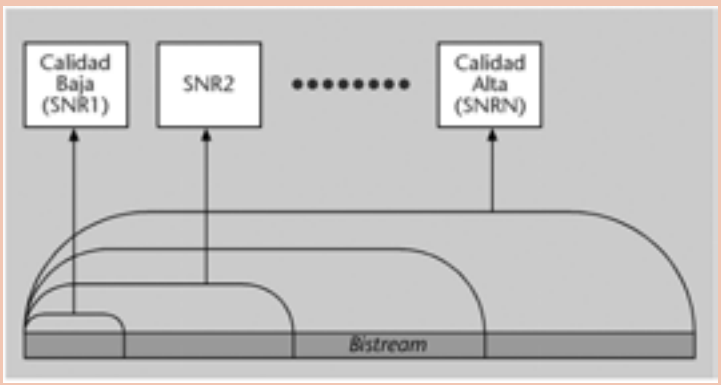
Escalabilidad espacial y SNR

El concepto de escalabilidad en la transmisión de imágenes está directamente relacionado con la transmisión progresiva. El nuevo estándar permite que el usuario pueda especificar que el *bitstream* disponga de los dos tipos de escalabilidad más importantes. La escalabilidad espacial se refiere a que el *bitstream* contiene distintas resoluciones de la imagen. La parte básica del *bitstream* proporciona una imagen de baja resolución (tamaño reducido) que progresivamente se va detallando hasta llegar a la resolución final. En la figura siguiente se representa de forma esquemática la estructura del *bitstream* con escalabilidad espacial.



Estructura básica de un *bitstream* escalable espacialmente

En la escalabilidad en SNR se transmite la imagen inicial a resolución completa, pero con baja calidad (sólo se cuantifican los coeficientes más importantes y con un error de cuantificación que puede ser considerable en estas primeras fases de la transmisión). La calidad de la imagen va mejorando progresivamente a medida que se transmiten los coeficientes adicionales y se aproximan mejor los primeros coeficientes. La filosofía básica de esta estructura del *bitstream* se representa esquemáticamente en la figura.



Estructura básica de un *bitstream* escalable espacialmente

La transformada wavelet proporciona, como veremos, diferentes propiedades que facilitan la codificación de la imagen con los dos tipos de escalabilidad. Evidentemente, el estándar también admite que en un mismo *bitstream* se utilicen los dos tipos de escalabilidad.

Estos tipos de escalabilidad permiten reproducir las imágenes utilizando distintas resoluciones y calidades que pueden depender del tipo de dispositivo de representación gráfica utilizado. Es especialmente interesante para utilizar en aplicaciones cliente–servidor de acceso lento ya que, en función de las características del dispositivo que se conecta, se puede proporcionar sólo el flujo de datos necesario. Así, la trama de datos que se envía a un dispositivo móvil puede consistir sólo en una primera aproximación a la imagen final, mientras que cuando se conecta un *display* de alta resolución se puede proporcionar toda la información disponible.

Codificación de regiones de interés

En algunas imágenes existen partes que son más importantes que otras. El estándar permite que el usuario especifique regiones de interés que se codifican con mayor calidad y menor distorsión que el resto de la imagen. Las regiones de interés son especialmente útiles en cámaras de transmisión de imágenes remotas en las que sólo una parte de la imagen contiene la información deseada (p.ej. vídeo-vigilancia, la región de interés son las puertas de acceso).

Acceso aleatorio a una parte de la imagen

Algunas partes de la secuencia de bits (previamente definidas por el usuario como regiones de interés) pueden ser accedidas directamente por el decodificador sin necesidad de procesar el resto de la secuencia.

Arquitectura abierta

La arquitectura del conjunto codificador–decodificador es abierta para optimizar el sistema para distintos tipos de aplicaciones específicas. De acuerdo con esta caracte-

rística, el decodificador debe ser capaz de implementar todas las técnicas de descompresión del núcleo del estándar y disponer de un analizador de la trama de bits. Si es necesario, el decodificador puede solicitar nuevos algoritmos para interpretar un modo de compresión que serían enviados desde la fuente (el codificador).

Combinación de imágenes bi-tonales con imágenes multinivel

Uno de los principales problemas del JPEG son los pobres resultados que se obtienen al comprimir imágenes bi-tonales (texto, gráficos) o imágenes sintetizadas por ordenador. El JPEG2000 debe tratar con imágenes con múltiples componentes y múltiples márgenes dinámicos (desde 1 bit hasta 16 bits por cada componente de color). Con esta característica es posible utilizar un único codificador para comprimir imágenes que combinen textos y gráficos con imágenes naturales o imágenes sintéticas generadas por ordenador.

Robustez frente a errores de transmisión

La trama de bits está diseñada para disponer de cierta capacidad de recuperarse respecto a errores de transmisión y permite que se apliquen capas adicionales de protección y corrección de errores. Algunas componentes o partes de la trama de datos están especialmente diseñadas para resincronizar el decodificador y evitar la existencia de errores graves en las zonas importantes de la imagen.

Descripción de alto nivel basada en contenido

La indexación y búsqueda de contenidos multimedia en archivos de datos es cada vez más importante. Actualmente se están desarrollando estándares específicos como el MPEG-7 para proporcionar soporte a este tipo de problemas. El JPEG2000 contempla la inserción de información de metadatos como parte del estándar de compresión.

La transformada wavelet: una visión general

La presentación rigurosa de los principios de la transformada wavelet, sus propiedades y su aplicación a la compresión de imagen requiere un análisis complejo que escapa de los propósitos de este texto. Nuestra presentación en los siguientes apartados carecerá de rigurosidad matemática y únicamente pretende que el lector tenga una idea aproximada de los procedimientos involucrados en el cálculo de la transformada wavelet y unas nociones básicas sobre las que se basan los compresores de imagen que utilizan este tipo de descomposición.

La transformada wavelet separa la imagen en componentes que tienen una interpretación frecuencial sencilla, a través de filtros paso alto y paso bajo como los estudia-

dos en el curso anterior. La imagen se codifica teniendo en cuenta los resultados que ha proporcionado su filtrado en las regiones de alta y baja frecuencia, y se puede recuperar posteriormente la imagen original a partir de estos datos.

Descomposición de una imagen mediante un banco de filtros

En la figura se muestra el esquema básico de una descomposición wavelet de 1 nivel para imágenes. La imagen se filtra mediante filtros paso bajo y paso alto elementales en las direcciones vertical y horizontal. En la primera etapa se realizan los dos filtrados complementarios (paso bajo y paso alto) en la dirección horizontal, y se obtienen las imágenes intermedias L-hor y H-hor (*Low* horizontal, *High* Horizontal). Con posterioridad, para cada una de las imágenes resultantes se aplican dos filtros complementarios en la dirección vertical. Como resultado de este proceso se obtienen 4 imágenes que se denominan LL (*Low*, *Low*), LH (*Low*, *High*), HL (*High*, *Low*) y HH (*High*, *High*).

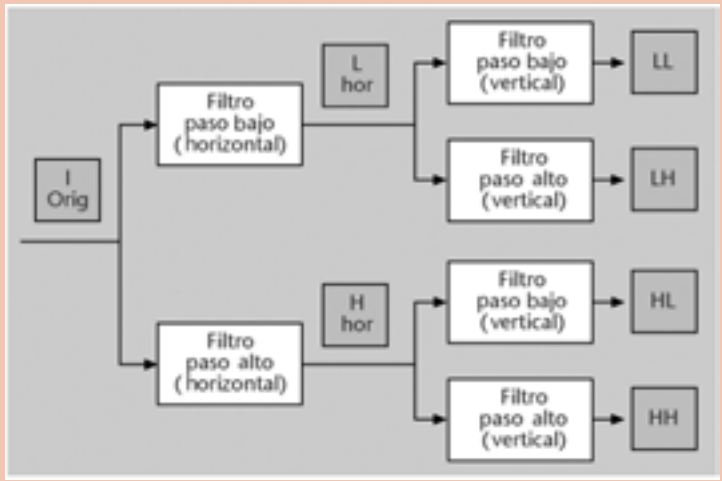


Diagrama básico de filtrado en la descomposición wavelet de una etapa

Tipos de filtros

Los filtros que se utilizan en la descomposición wavelet deben cumplir unas propiedades básicas para garantizar que la transformada puede ser invertida y que, por tanto, es posible recuperar la imagen original a partir de las descomposiciones. Evidentemente, los filtros pueden expresarse en forma de plantillas, tal y como habíamos estudiado en el curso anterior. Los filtros más sencillos con los que puede realizarse una transformación wavelet son los de Harr. Las plantillas de estos filtros se ilustran en la figura y puede expresarse de forma compacta como:

$$H_{Low} = [0.7071, 0.7071]$$
$$h_{High} = [0.7071, -0.7071]$$

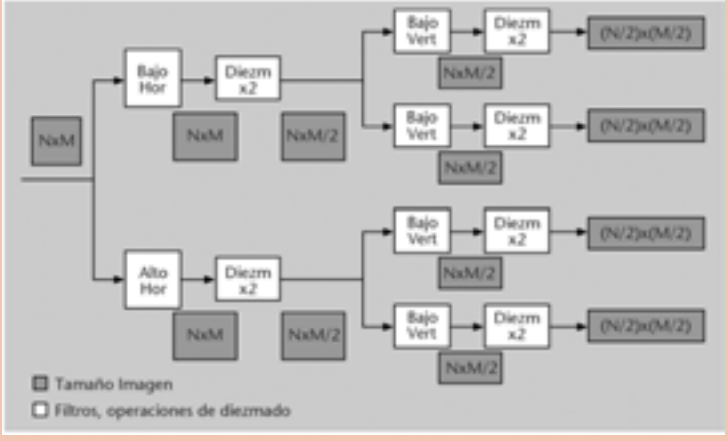
Reducción del número de muestras (diezmado)

Al aplicar filtros paso bajo y paso alto a una señal se reduce su ancho de banda, con lo que la frecuencia de muestreo original puede reducirse. En efecto, si consideramos el caso más sencillo de las señales unidimensionales (como la señal de audio), es fácil comprobar que un filtrado de la señal puede reducir el número de muestras sin que se produzca una pérdida de la información.

Si consideramos una señal de audio analógica con 20 KHz, el teorema del muestreo nos dice que la frecuencia mínima de muestreo para que no se produzca pérdida de información debe ser como mínimo de 40 KHz. Suponiendo que adquirimos esta señal de audio con una frecuencia de muestreo de 40 KHz y que posteriormente aplicamos un filtro paso bajo digital de 10 KHz, la nueva señal sólo contiene información relevante en la banda de 0 a 10 KHz (el resto suponemos que ha sido eliminado por el filtro digital). Esto significa que la señal resultante hubiera podido ser muestreada utilizando sólo la mitad de las muestras originales, es decir, tomando una frecuencia de muestreo de 20KHz.

Plantillas paso bajo y paso alto asociadas a los filtros de Haar en la transformada wavelet

El proceso de reducir la frecuencia de muestreo de la señal digital después del filtrado se conoce como *diezmado* (descarte de muestras). En nuestro ejemplo podemos eliminar la mitad de las muestras (una de cada dos muestras) sin que se produzca pérdida de información puesto que son las muestras que hubiéramos obtenido si se hubiese muestreado directamente una señal de audio filtrada de forma analógica a 10 KHz.



Esquema de filtrado y diezmado en la transformada wavelet

Con un razonamiento parecido es posible demostrar que al filtrar una imagen (señal en dos dimensiones) se reduce su ancho de banda y que, por lo tanto, es posible reducir su número de muestras sin que se produzca una pérdida de información.

La reducción del número de muestras puede realizarse en la misma dirección en la que se aplica el filtrado (horizontal o vertical) y en la forma en la que se ilustra en la siguiente figura. La frecuencia de corte de los filtros de Haar está diseñada para que la frecuencia de muestreo de la parte de baja frecuencia y la de alta frecuencia puedan reducirse a la mitad. La figura indica los tamaños de imagen que se obtienen en cada operación de diezmado (filas × columnas).

Para simplificar el diagrama de filtros horizontales y verticales y los módulos de reducción del número de muestras, la transformada wavelet puede representarse de forma comprimida tal y como se indica en la siguiente figura.

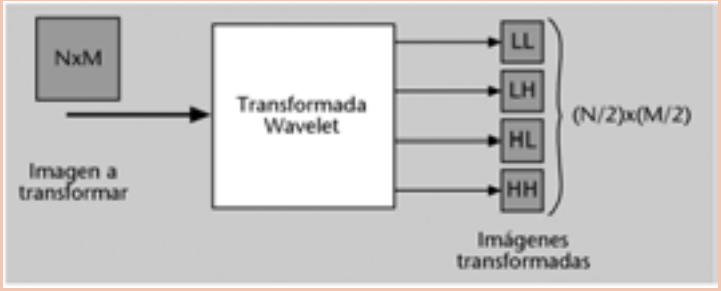


Diagrama de bloques simplificado de la transformada wavelet de un nivel de descomposición

Características de las imágenes resultantes

A partir del diagrama de bloques detallado de la transformada wavelet como un banco de filtros paso bajo y paso alto en las direcciones horizontales y verticales, resulta fácil interpretar los resultados que obtendremos en las 4 imágenes de salida.

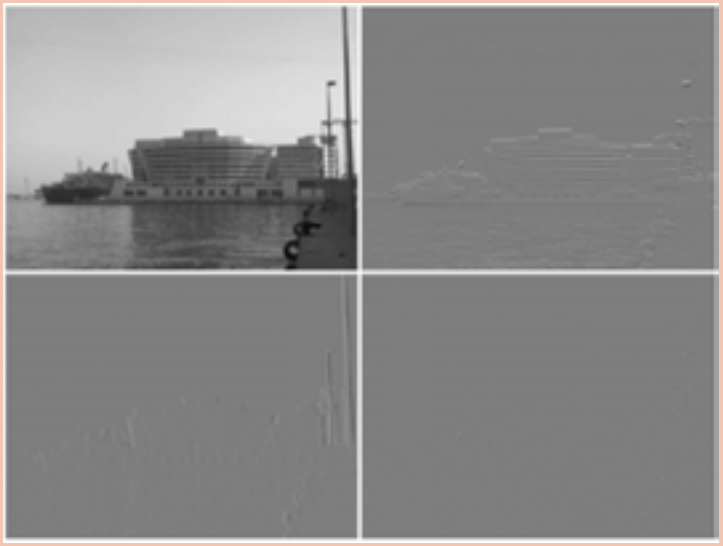
- Imagen LL. La imagen LL es la que se obtiene de filtrar paso bajo tanto en la dirección horizontal como en la vertical. Por ello, el contenido visual de la imagen será aproximadamente el mismo que el de la imagen original, pero con una menor resolución. En efecto, al filtrar paso bajo los contornos de la imagen, quedarán menos definidos. Al reducir el tamaño de la imagen en un factor 2 tanto en las filas como en las columnas, obtenemos una imagen con las mismas características que la original, pero de tamaño más pequeño.

- Imagen LH. La imagen LH se obtiene al aplicar un filtro paso bajo en el sentido horizontal más un filtro paso alto en el sentido vertical. El primer filtro reduce la definición horizontal de la imagen, mientras que el segundo enfatizará los contornos horizontales (realza los cambios rápidos en el sentido vertical). La imagen resultante será de menor tamaño y contendrá principalmente los contornos horizontales.

- Imagen HL. Es el caso contrario al LH. En este caso, la imagen contendrá fundamentalmente la información de contornos verticales.

- Imagen HH. Esta imagen se obtiene al aplicar un filtro paso alto tanto en la dirección horizontal como en la vertical. El resultado incluye todas las componentes de alta frecuencia, y se centra sobre todo en las componentes diagonales.

En la figura siguiente se muestra el resultado de las 4 componentes de salida. Las imágenes están ordenadas de izquierda a derecha y de arriba abajo según la secuencia LL, LH, HL y HH. La imagen de baja frecuencia toma siempre valores positivos, por lo que se representan directamente los niveles de gris normalizados respecto al máximo que se toma como blanco. El resto de las imágenes puede tomar valores negativos y positivos. Se han representado tomando como nivel de gris medio el valor nulo de manera que los valores negativos aparecerán como más oscuros y los positivos como claros.



Representación de las imágenes LL, LH, HL y HH de la transformada wavelet

Es importante observar que la descomposición frecuencial de las 4 imágenes concuerda con los resultados que se preveían. La componente HH contiene únicamente detalles de muy alta frecuencia que se concentran en la parte del mar, los edificios y el barco. Las imágenes LH y HL contienen respectivamente la información de contornos horizontales y verticales.

Transformada wavelet inversa

A partir de las 4 imágenes resultantes de la transformada wavelet es posible recomponer, a resolución completa, toda la información de la imagen original. La transformada wavelet inversa puede representarse esquemáticamente como el siguiente bloque de transformación.

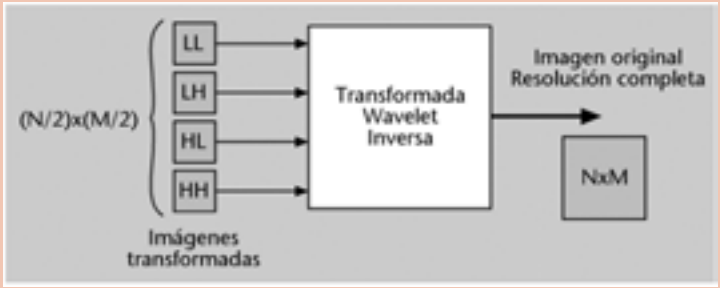


Diagrama global de la transformada wavelet inversa

Detalle del tratamiento de señal en la transformada inversa

El esquema detallado de la transformada inversa como un banco de filtros es muy parecido al de la transformada directa y se representa en la figura siguiente. La transformada directa suele denominarse *banco de filtros de análisis* (descomposición de la imagen en partes) y la transformada inversa *filtros de síntesis*. Cuando se utilizan filtros de Haar para el análisis, pueden usarse los mismos filtros para la parte de síntesis. Para otro tipo de filtros es posible que los coeficientes de los filtros de análisis y los de síntesis no coincidan.

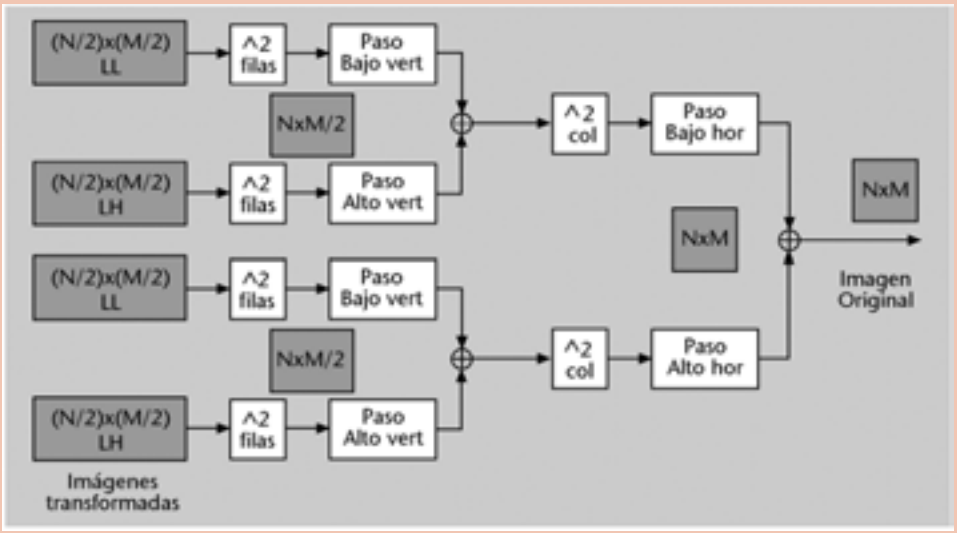


Diagrama del banco de filtros asociado a la transformada wavelet inversa

El banco de filtros de síntesis es totalmente equivalente al banco de análisis, exceptuando que en los bloques en los que se realizaba la reducción del número de filas y/o columnas ahora se intercalan ceros. El proceso consiste en intercalar filas o columnas de ceros entre las filas o columnas reales de la imagen. El proceso de filtrado es el que se encarga de sustituir los ceros que hemos añadido por los coeficientes de luminancia y/o croma que les corresponde a estas posiciones. Como resultado final se obtiene una imagen completa (NxM) que se corresponde con la imagen original.

Codificación de la imagen en el dominio transformado

La transformada wavelet y su antitransformada puede aprovecharse para realizar la codificación de la imagen. La idea básica consiste en transformar una imagen de $N \times M$ píxeles para obtener 4 imágenes de $(N/2) \times (M/2)$ píxeles que son las que realmente se codifican, almacenan o transmiten. Hemos visto que por medio de la transformada inversa podemos recuperar la imagen original a partir de las 4 imágenes reducidas, por lo que, de entrada, el proceso descrito es totalmente lícito.

La primera pregunta que se plantea es qué ventaja se obtiene al codificar las 4 imágenes transformadas en vez de la imagen original. Una primera inspección del problema nos dice que el número total de elementos de imagen que debemos codificar no ha cambiado. En efecto, estamos proponiendo almacenar 4 imágenes con $(N/2) \times (M/2)$ elementos de imagen, con lo que el número total de elementos de imagen es $N \times M$ y coincide con los elementos de la imagen original.

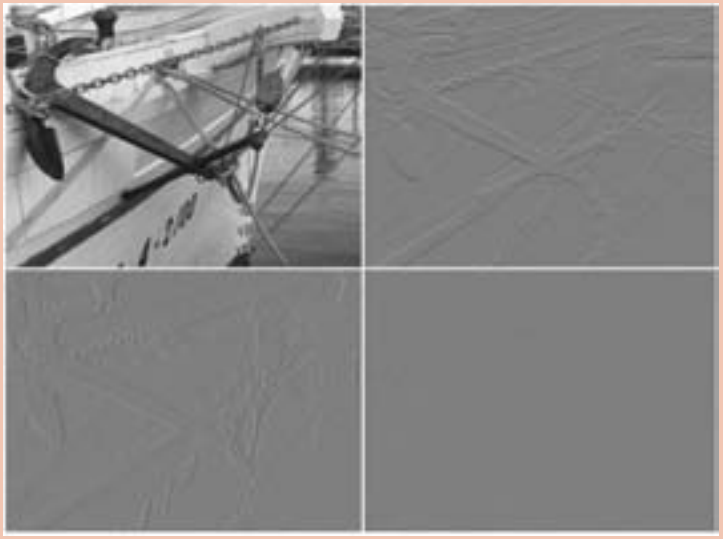
De hecho, el problema es parecido al de la codificación mediante la transformada coseno. En aquel caso se toman bloques de 8×8 elementos de imagen y se transformaban obteniendo 64 coeficientes transformados. La ventaja principal es que de los 64 coeficientes obtenidos, la mayor parte era nula y no resultaba necesario almacenarlos o transmitirlos para recuperar la imagen original con exactitud.

En el caso de la transformada wavelet ocurre algo parecido. Las componentes de alta frecuencia HL, LH y HH son prácticamente nulas en la mayoría de sus elementos de imagen (en la transformada del ejemplo anterior prácticamente todas las regiones tienen un nivel de gris medio que se corresponde con el nivel cero).

Sólo resultan visibles las zonas en las que la imagen original tiene contornos verticales, horizontales o, en general, componentes de alta frecuencia. En la mayoría de las imágenes naturales esto ocurre sólo en un porcentaje de píxeles muy pequeño comparado con el área total de la imagen. Utilizando esta característica es posible codificar estas tres componentes de la transformada wavelet de forma muy eficiente, utilizando pocos bits, de manera que se puede obtener una compresión eficaz al almacenar las 4 imágenes resultantes en vez de la imagen original.

La imagen de baja frecuencia tiene un contenido parecido al de la imagen original, aunque es de un tamaño menor. Su codificación sigue siendo compleja y, en principio, deberemos invertir la mayor parte de los bits en su codificación.

En la figura siguiente se muestra un nuevo ejemplo de la descomposición de una imagen mediante la transformada wavelet. En este caso, la imagen tiene una predominancia de contornos diagonales, por lo que aparecen componentes significativas en las tres imágenes de alta frecuencia. No obstante, de nuevo estos coeficientes significativos sólo se producen en un número de píxeles muy reducido dentro de todos los píxeles de la imagen, por lo que podrá realizarse una codificación eficiente utilizando técnicas ya conocidas como la codificación por longitud de ceros, los códigos de Huffman o los códigos aritméticos.



Ejemplo de descomposición de una imagen en las 4 componentes de la transformada wavelet

Codificación con pérdidas y sin pérdidas

La transformada wavelet y su inversa son en principio sin pérdidas. Las pérdidas se introducen cuando se aproximan los coeficientes transformados que, en este caso, son las imágenes que se obtienen una vez realizada la descomposición. Si las imágenes LL, LH, HL y HH se almacenan sin pérdidas, la imagen original puede recuperarse de forma exacta. No obstante, cuando las imágenes resultantes se cuantifican con pasos de cuantificación elevados, la imagen original sólo podrá recuperarse de forma aproximada.

Es importante observar que en el caso en que se desee realizar una codificación con pérdidas, la imagen de baja frecuencia es la que debe guardarse con mayor precisión ya que es la que contiene la información más importante para la reconstrucción de la imagen original. El resto de las imágenes cuya información se corresponde con los contornos puede codificarse con menor precisión aprovechando que el sistema visual humano no es muy sensible a los errores de cuantificación en estas regiones. Efectivamente, cuando aparece un contorno en una imagen, es importante que esté bien definido en el espacio (para no producir desenfoque), pero no es tan importante que el nivel de cambio esté perfectamente cuantificado. El sistema visual no podrá apreciar diferencias entre un contorno real en el que se produce un cambio de nivel de gris de 72 unidades a un contorno reconstruido en el que el cambio de nivel de gris es de 80 ó 90 unidades (aproximación).

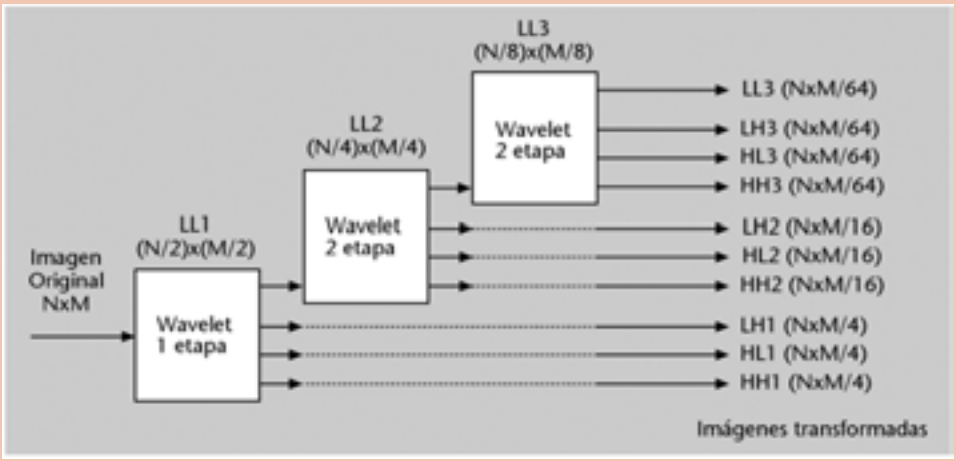
Descomposición wavelet de varios niveles

El resultado de la descomposición nos produce 4 imágenes de tamaño reducido. Las tres con componentes de alta frecuencia pueden ser codificadas con métodos eficientes debido a que tienen un elevado número de coeficientes próximos a cero. En cambio, la imagen de baja frecuencia presenta una información altamente correlada con

la original y, aunque es de menor tamaño, resulta en principio compleja de comprimir debido a que sus niveles de gris están uniformemente distribuidos en todo el margen dinámico.

Una estrategia más que evidente para comprimir de nuevo esta imagen de baja frecuencia (LL) es volver a aplicar la descomposición wavelet. Con ello, la imagen se volverá a descomponer en 4 subimágenes, tres de ellas de alta frecuencia que pueden ser comprimidas con facilidad y una nueva imagen de baja frecuencia con un tamaño reducido. Al aplicar recursivamente la transformada wavelet sobre la componente de baja frecuencia (LL) podemos ir obteniendo diferentes imágenes de alta frecuencia y una de baja frecuencia de tamaño cada vez más pequeño y que, por lo tanto, ocupará un número menor de bits.

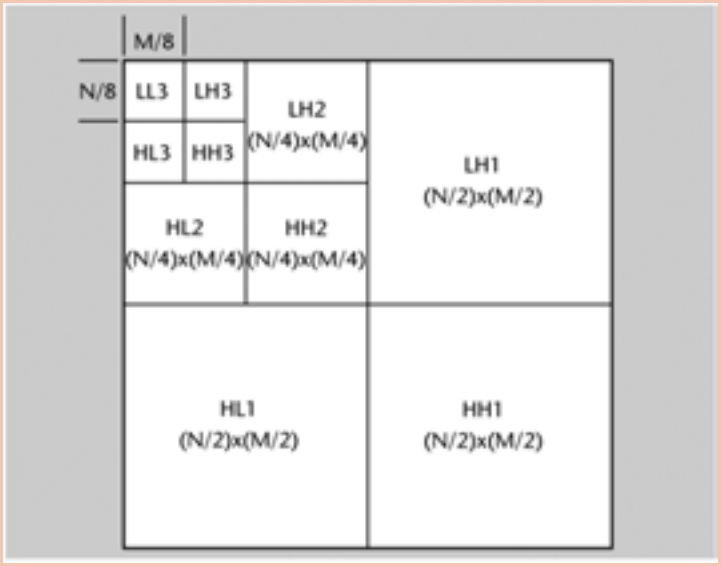
En la figura siguiente se representa un esquema básico de descomposición de una imagen en varios niveles de transformada wavelet (3 etapas de descomposición). El bloque básico es el mismo que habíamos representado para la transformada wavelet de un solo nivel de descomposición que se aplica exclusivamente sobre la componente de baja frecuencia.



Ejemplo de descomposición de 3 niveles de una imagen

Ejemplo de descomposición de una imagen en tres niveles

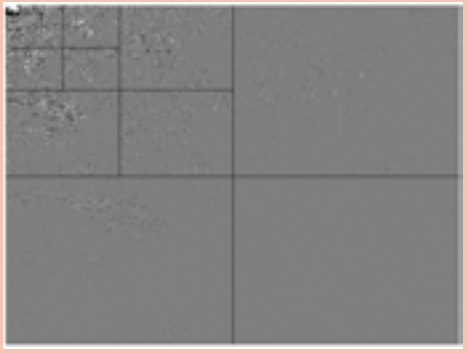
A medida que se van realizando etapas de la descomposición wavelet, la imagen se subdivide en un número mitad de filas y columnas, por lo que las imágenes resultantes de los niveles de descomposición altos tienen un tamaño menor que las de los primeros niveles. La representación gráfica de las distintas imágenes obtenidas en cada nivel de descomposición suelen representarse en una matriz con un tamaño de píxeles igual al de la imagen original y con una distribución como la que se muestra en la figura.



Distribución de las imágenes de los distintos niveles de descomposición en la representación de una transformada wavelet multinivel

En la figura siguiente se muestra el resultado de una transformada wavelet de 5 niveles aplicado sobre una imagen real. Observad que la imagen LL5 resulta de un tamaño muy pequeño, por lo cual serán necesarios muy pocos bits para codificarla correctamente. También se proporciona la imagen original para tener una referencia de su contenido visual.

Imagen original sobre la que se aplica la descomposición wavelet de 5 niveles



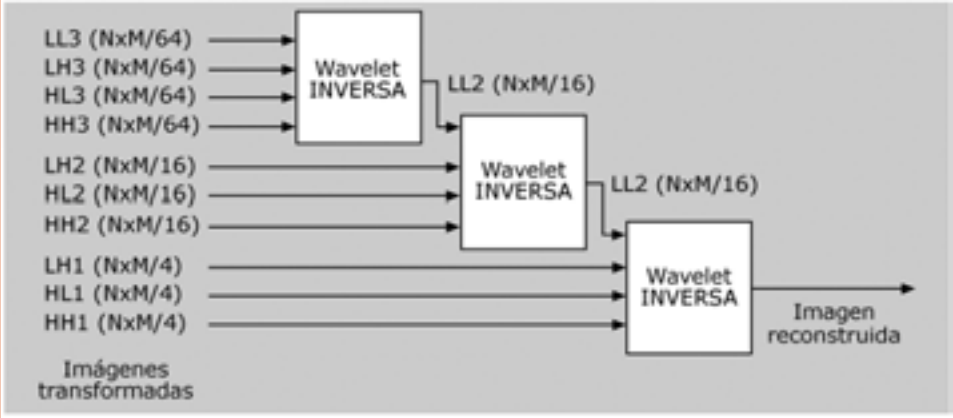
Descomposición wavelet de 5 niveles de la imagen anterior

Reconstrucción de la imagen original a partir de las descomposiciones multinivel

La reconstrucción de la imagen original se realiza de forma progresiva a partir de los niveles de descomposición más elevados. En efecto, a partir de las componentes de baja y alta frecuencia del nivel más alto (p.ej. LL5, lH5, HL5 y HH5 en el ejemplo anterior) podemos reconstruir la imagen de baja frecuencia asociada al nivel inmediatamente inferior, es decir, LL4. A partir de esta imagen y las restantes de alta frecuencia (LH4, HL4 y HH4) podemos reconstruir LL3 y así sucesivamente. En la siguiente figura se representa el proceso de reconstrucción utilizando bloques de la transformada inversa.

Es importante observar que el proceso de reconstrucción natural es progresivo y sigue un esquema paralelo al de la escalabilidad espacial. En efecto, la primera imagen necesaria para iniciar el proceso de reconstrucción es la de baja frecuencia que está asociada al nivel de descomposición más alto. Es la imagen de tamaño más pequeño que contiene una información visual con un contenido equivalente (pero con baja resolución) al de la imagen original. A partir de esta imagen, junto con las de alta frecuencia, es posible reconstruir la imagen de baja frecuencia del nivel inferior, que tendrá una mayor resolución. Las imágenes de baja frecuencia que se van reconstruyendo aumentan progresivamente su resolución hasta llegar a la resolución completa de la imagen original. Por tanto, el propio proceso de reconstrucción induce directamente a un sistema de decodificación escalable.

La escalabilidad en SNR (calidad) también es bastante directa, ya que depende de la precisión con la que se codifican los coeficientes de alta frecuencia. Si en un primer nivel (capa base) se realiza una cuantificación aproximada de los coeficientes y se transmiten en primer lugar al decodificador, éste podrá obtener una primera aproximación de baja calidad a la imagen original. Cabe notar que esta primera aproximación puede obtenerse directamente mediante resoluciones que aumentan progresivamente. Posteriormente, si se envía un refinamiento de la cuantificación previa de los coeficientes de alta frecuencia, podremos mejorar la calidad de la imagen decodificada. El esquema de cuantificación del JPEG2000 establece un orden en la transmisión de los píxeles de alta frecuencia en función de su contenido. Se transmiten en primer lugar los que son más importantes. De esta manera, los primeros bits que se envían al decodificador contienen la información más significativa que permite recomponer una primera versión aproximada con un número muy reducido de bits. Examinaremos con algo más de detalle esta estrategia de codificación posteriormente.



Esquema de reconstrucción de imágenes en la transformada inversa multinivel

Resumen de características de la transformada wavelet

En este apartado resumimos las principales características de la transformada wavelet para su aplicación en la compresión de imagen:

- **Descomposición en bandas frecuenciales de la información**. Cada etapa de la transformada wavelet obtiene 4 imágenes de tamaño reducido. Tres de estas imá-

genes corresponden a información de alta frecuencia que puede comprimirse de forma muy eficiente utilizando códigos de longitud variable o códigos aritméticos.

- **Interpretación de las componentes de alta frecuencia.** Las componentes de alta frecuencia contienen información sobre los contornos horizontales, verticales y diagonales de la imagen. El resto de la información es prácticamente nulo. Podemos pensar que la compresión de la imagen está basada en transmitir la información de baja frecuencia con menor resolución y posteriormente añadir la información sobre los contornos y detalles de alta frecuencia de la imagen (componentes LH, HL y HH).

- **Descomposición multinivel de la imagen.** Aplicando recursivamente la transformada wavelet a la parte de baja frecuencia de la imagen es posible obtener descomposiciones en partes de baja y alta frecuencia más finas. Con ello la imagen de baja frecuencia puede reducirse a un tamaño suficientemente pequeño como para que pueda ser codificada con precisión utilizando pocos bits. Generalmente 4 niveles de descomposición resultan suficientes para obtener una excelente capacidad de compresión.

- **Reconstrucción progresiva natural.** El esquema de reconstrucción exige recuperar previamente las partes de baja frecuencia de los sucesivos niveles de descomposición. Esto permite que pueda aplicarse una reconstrucción escalable espacialmente. La escalabilidad en SNR puede conseguirse cuantificando los coeficientes con menor precisión en la capa base e ir aumentado la calidad de la cuantificación progresivamente.

- **Desaparición del efecto de bloque de la DCT.** Debido a que la transformada se aplica sobre una imagen de gran tamaño no aparece el efecto de bloque característico de la DCT. No obstante, cuando se reduce mucho la tasa de transmisión, la incorrecta codificación de los contornos de alta frecuencia puede producir efectos de "oscilación" del nivel de gris en las transiciones entre objetos. No obstante, puede aumentarse la tasa de compresión más allá de lo que tolera la DCT sin que se aprecien efectos de degradación visualmente molestos.

Codificación aritmética

La codificación aritmética es una alternativa a la codificación mediante códigos de longitud variable que se utiliza en distintos métodos de compresión de imágenes estáticas y vídeo. El algoritmo JPEG convencional utiliza una variante de esta forma de codificación en uno de sus modos extendidos. También se utiliza esta estrategia de codificación en el JPEG2000 y en otros codificadores avanzados orientados a videoconferencia (H.263+ y MPEG-4).

En este apartado veremos brevemente la idea general de la codificación aritmética que, como los códigos de Huffman, también está basada en reducir la tasa de bits aprovechando el análisis estadístico de las frecuencias o probabilidades con las que

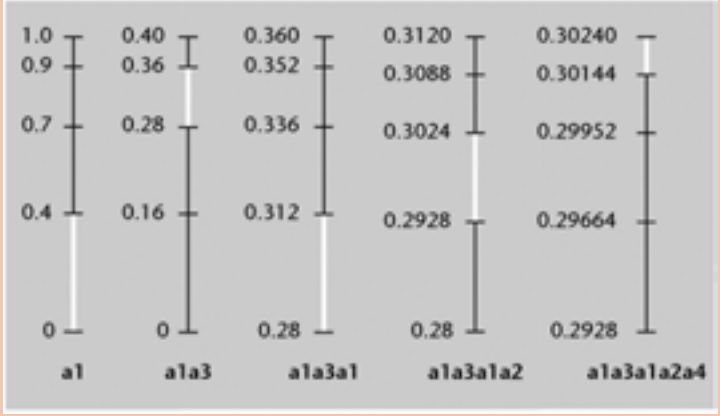
se producen los símbolos. La codificación aritmética puede considerarse como otra alternativa para realizar la codificación de los símbolos basada en el estudio de la entropía de la fuente. Existen muchas variantes de la codificación aritmética cuyo estudio exhaustivo escapa de los propósitos de este texto y, por lo tanto, sólo consideramos sus aspectos e ideas más generales.

Principios generales

La codificación aritmética se basa en asociar números reales comprendidos entre 0 y 1 a secuencias o cadenas de mensajes de la fuente. Aunque en la práctica es difícil de implementar, suele conseguir resultados de compresión excelentes, superiores, aproximadamente en un 25% de eficiencia de compresión, a la codificación de Huffman. Recordemos que Huffman era un código óptimo sólo cuando se restringía que cada mensaje de la fuente debía ser codificado con un símbolo. En el caso de la codificación aritmética se consiguen mejores resultados debido a que los símbolos corresponden a cadenas de mensajes.

Existen distintas variantes de los códigos aritméticos, sobre todo en lo que respecta a su implementación y los formatos de representación de los números reales con aritmética finita. Esencialmente, el proceso de codificación se ilustra en la figura adjunta para la siguiente cadena de mensajes: $a_1a_3a_1a_2a_4$. Para simplificar el procedimiento, suponemos que los mensajes proceden de una fuente con 4 mensajes posibles a_1, a_2, a_3 y a_4 con probabilidades 0,4; 0,3; 0,2 y 0,1. El mensaje a_4 se utiliza exclusivamente para indicar el fin de cadena. El número real asignado al mensaje se va determinando progresivamente a medida que se codifican los distintos elementos de la cadena.

Para codificar el primer elemento, el intervalo real [0; 1) se divide en 4 porciones proporcionales a la probabilidad de cada uno de los mensajes. El mensaje a_1 quedará codificado como el intervalo real [0; 0,4), el mensaje a_2 como el intervalo [0,4; 0,7) y así sucesivamente. En la figura se indica el intervalo en el que ha sido codificado el primer elemento de la cadena. Para codificar el segundo elemento, tomamos el intervalo definido por el primero y se vuelve a dividir en 4 intervalos con una longitud proporcional a sus probabilidades, aplicando progresivamente este procedimiento a medida que se incorporan más elementos de la cadena.



Ejemplo de codificación de una cadena de mensajes mediante códigos aritméticos

En la gráfica se observa que la secuencia que pretendíamos codificar quedaría asignada al intervalo [0,30144; 0,30240). Cualquier número real dentro de este intervalo, por ejemplo el 0,302, codificaría de forma unívoca el mensaje. En este ejemplo, es suficiente con tres cifras decimales para codificar la cadena de 5 mensajes, lo que representa una codificación bastante eficiente de la fuente. Notad que los códigos resultantes son de longitud variable, puesto que en función de la cadena de mensajes podemos necesitar más o menos dígitos para codificar el mensaje. La implementación del codificador y del decodificador en tiempo real o mediante *software* no son evidentes y en la práctica existen distintas estrategias para optimizar los tiempos de cálculo.

Arquitectura general del codificador JPEG2000

En este apartado examinaremos los principales elementos del codificador JPEG2000 desde un punto de vista sistemático. Los detalles del procedimiento de cuantificación de las imágenes, la estrategia de composición de la tramas de bits escalables y la definición de las regiones de interés son muy complejas y sus detalles se omitirán. Pretendemos únicamente que el lector tenga una idea general del algoritmo, sus ventajas respecto al JPEG convencional y sus posibles limitaciones.

Etapas básicas del proceso de codificación

El diagrama de bloques básico del sistema es el mismo que el de cualquier sistema de compresión basado en el uso de transformadas. En la parte del codificador se realiza la transformación de la imagen (en este caso, utilizando la transformada wavelet), posteriormente se realiza la cuantificación de los coeficientes resultantes (las imágenes resultantes de la transformación) y finalmente se utiliza un codificador entrópico (basado en la codificación aritmética) para formar el *bitstream* definitivo que contendrá la información.

El decodificador realiza las mismas etapas en el sentido inverso: primero se realiza la decodificación aritmética, posteriormente se recuantifican los valores de los coeficientes y finalmente se reconstruye la imagen utilizando la transformada wavelet inversa. En los siguiente apartados definiremos brevemente las particularidades de cada uno de los módulos. En la figura siguiente se representa un diagrama con los bloques principales del proceso de codificación y decodificación.

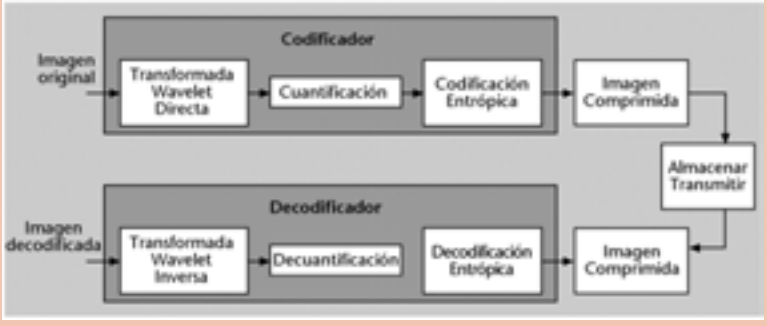


Diagrama de bloques del conjunto codificador–decodificador JPEG2000

Descomposición de la imagen

Antes de aplicar la transformada wavelet, la imagen se descompone en un mosaico de imágenes. El mosaico está formado por subimágenes rectangulares, todas del mismo tamaño, que no se superponen. La transformada wavelet se aplicará de forma independiente a cada una de las subimágenes que se codificarán por separado. Esta descomposición en mosaico es útil sobre todo para imágenes de gran tamaño, ya que de este modo pueden reducirse los requisitos de memoria del codificador y del decodificador, realizando la decodificación de cada una de las partes por separado. El tiempo de cómputo de la wavelet también se reduce considerablemente cuando se reduce el tamaño de la transformación de forma que resulta menos costoso realizar N transformadas reducidas que una única transformada con un número de elementos N veces mayor. Cuando se trabaja con imágenes de varias componentes (p.ej. imágenes en color) que tienen tamaños distintos (p.ej. YUV 4:2:0), las componentes de baja resolución deben remuestrearse para que todas ellas tengan tamaños consistentes con los elementos del mosaico.

Si el tamaño de la imagen no es un múltiplo entero del tamaño de cada una de las subimágenes, se incluyen columnas y filas de ceros en la parte derecha e inferior de la imagen. Todas las operaciones de transformación, cuantificación y codificación se realizan de forma independiente para cada uno de los mosaicos. Por tanto, la codificación en mosaicos permite reconstruir diferentes partes de la imagen de forma independiente, y se acepta la decodificación parcial de una imagen.

Los tamaños de cada uno de los mosaicos son definidos por la propia aplicación de codificación o por el usuario, y se admite que toda la imagen esté codificada como un todo. Los resultados obtenidos mejoran a medida que el tamaño de la subimagen aumenta, y se obtiene el caso óptimo cuando toda la imagen se codifica como un todo. A medida que se aumenta el número de subimágenes en las que se realiza la descomposición, aparecen errores de reconstrucción en los límites de las imágenes que para tasas de codificación muy bajas pueden resultar molestos.

En la tabla siguiente se presentan los resultados de PSNR típicos que se obtienen al codificar una imagen de 2048x2560 píxeles con una tasa de 0.0625 bpp utilizando distintos tamaños de subimagen. Estos resultados pretenden proporcionar una idea de cómo se degrada la calidad de la codificación al aumentar el número de elementos del mosaico. En última instancia, es el decodificador el que debe valorar el número de componentes en el que puede descomponerse una imagen teniendo en cuenta su tamaño total, los requerimientos de memoria y capacidad de proceso del decodificador y las necesidades de acceder a partes distintas de la imagen de forma independiente.

Resultados comparativos al codificar una imagen JPEG2000 con diferentes tamaños de subimagen

Imagen	PSNR (sin transf. color / con transf. color)
Sin descomposición en subimágenes	23,5 / 25,07
Subimágenes de tamaño 256 × 256	23,26 / 24,70
Subimágenes de tamaño 128 × 128	22,80 / 23,91

Transformaciones de color

La transformada wavelet puede aplicarse directamente sobre cada una de las componentes de la imagen. El algoritmo es suficientemente general como para permitir la codificación de imágenes que están expresadas en un número de componentes elevado. Estas imágenes pueden proceder de sistemas de exploración vía satélite en los que además de las componentes del espectro visible (RGB) se pueden adquirir componentes correspondientes a las bandas del infrarrojo, ultravioleta, etc.

Cuando la imagen se proporciona en componentes RGB, suele resultar conveniente realizar una transformación previa a componentes YCrCb ya que, en general, se obtienen mejores resultados. No obstante, esta transformación es opcional y no está impuesta por el estándar. En cualquier caso, la transformada wavelet se aplicará de forma independiente a cada una de las componentes resultantes.

La mejora en resultados obtenida cuando se realizan las transformaciones de componentes puede observarse en la tabla anterior donde se presentan los resultados de PSNR obtenidos al aplicar la codificación directamente sobre las componentes RGB o sobre las componentes YCrCb (con o sin transformación de color).

El estándar define dos posibles transformaciones de componentes RGB aYCrCb. La primera es la conversión habitual entre los dos tipos de componentes que utiliza números reales para realizar la transformación. Esta transformación sólo puede aplicarse cuando se realiza una codificación con pérdidas, ya que durante la etapa de cuantificación los valores transformados deben aproximarse por números enteros y ya no será posible recuperar los valores originales de las componentes de la imagen. Esta transformación se denomina ICT (*Irreversible Color Transform*) y está definida a partir de las siguientes relaciones matriciales, que coinciden numéricamente con las presentadas en la primera etapa exceptuando que, en este caso, no es necesario realizar la normalización de las componentes Cr y Cb en el margen de 0 a 255:

Transformación directa

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ C_b &= \text{ñ}\, 0.16875R \,\text{ñ}\, 0.33126G + 0.5B \\ C_r &= 0.5R \,\text{ñ}\, 0.41869G \,\text{ñ}\, 0.08131B \end{aligned}$$

Transformación inversa

$$\begin{aligned} R &= Y + 0.1402C_r \\ G &= Y \,\text{ñ}\, 0.34413C_b \,\text{ñ}\, 0.71414C_r \\ B &= Y + 1.772C_b \end{aligned}$$

Alternativamente puede utilizarse una transformación de color reversible (RCT – *Reversible Color Transformation*) que aproxima las transformaciones anteriores utilizando únicamente valores enteros. Las matrices asociadas a esta transformación vienen dadas por:

Transformación directa

$$Y = \frac{R + 2G + B}{4}$$
$$U = R \tilde{n} G$$
$$V = B \tilde{n} G$$

Transformación inversa

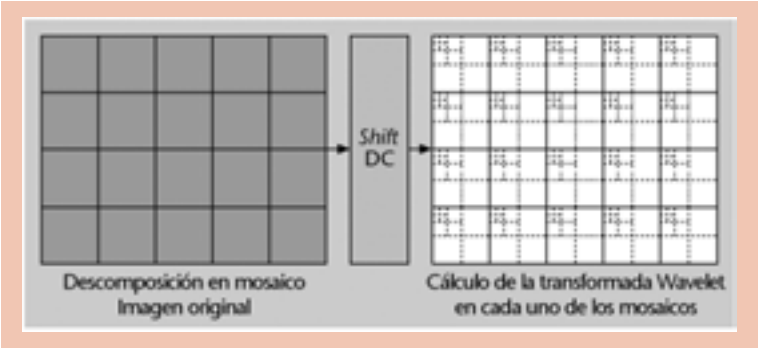
$$G = Y \tilde{n} \left(\frac{U + V}{4} \right)$$
$$R = U + G$$
$$B = V + G$$

La ventaja de esta transformación alternativa es que después de realizar la codificación sin pérdidas todavía es posible recuperar las componentes originales en las que se proporcionaba la imagen. Así, si la imagen original está codificada en RGB y se convierte a YCbCr mediante la transformación RCT, codificando el resultado sin pérdidas es posible recuperar los valores originales de RGB a partir de los YCbCr que obtendrá el decodificador. En cambio, si utilizamos la transformación ICT, esto no será posible. La transformación RCT puede utilizarse tanto cuando el JPEG2000 trabaja en modo pérdidas o sin pérdidas. Evidentemente, la transformación ICT sólo puede utilizarse cuando se está utilizando el modo con pérdidas. La calidad final que se obtiene en modo pérdidas es superior cuando se utiliza la conversión ICT, lo que justifica su utilización.

Cálculo de la transformada wavelet

Antes de aplicar la transformada wavelet se realiza la extracción de la componente de continua de las componentes de la imagen. Evidentemente, sólo se aplica en las componentes que tienen un nivel medio distinto de cero debido a que se representan mediante números positivos (la luminancia y las señales R, G y B siempre son positivas; en cambio Cr y Cb ya no tienen componente de continua).

En la figura siguiente se representa el esquema general de descomposición de la imagen en mosaico, la extracción de la componente continua y el cálculo de la transformada wavelet. En este esquema hemos supuesto que no se aplica ninguna transformación de componentes de color.



Los coeficientes de los filtros de la transformada wavelet pueden ser de dos tipos distintos. El primer filtro se utiliza cuando se realiza codificación con pérdidas y se denomina *filtro de Daubechies*. En este caso, al contrario que con los filtros de Harr, el filtro de análisis y el filtro de síntesis son distintos y los de alta y baja frecuencia tienen un número de coeficientes diferente. En la tabla siguiente se proporcionan los coeficientes de los filtros.

Filtro Daubechies de análisis (transformada wavelet directa)		
Coeficiente	**Paso bajo**	**Paso alto**
–4	0.026748	
–3	–0.01686	0.09127
–2	–0.07822	–0.05754
–1	0.26684	–0.59127
0	0.60295	1.11508
1	0.26684	–0.59127
2	–0.07822	–0.05754
3	–0.01686	0.09127
4	0.026748	

Filtro Daubechies de síntesis (transformada wavelet inversa)		
Coeficiente	**Paso bajo**	**Paso alto**
–4		0.026748
–3	–0.09127	0.01686
–2	–0.05754	–0.07822
–1	0.59127	–0.26684
0	1.11508	0.60295
1	0.59127	–0.26684
2	–0.05754	–0.07822
3	–0.09127	0.01686
4		0.026748

Cuando se debe realizar la codificación sin pérdidas, se utilizan unos filtros con coeficientes racionales que permiten mantener la precisión completa en los cálculos intermedios. Los coeficientes utilizados se representan en la tabla siguiente:

	Filtro de análisis		**Filtro de análisis**	
Coeficientes	**Paso bajo**	**Paso alto**	**Paso bajo**	**Paso alto**
–2	–1/8			–1/8
–1	2/8	–1/2	1/2	–2/8
0	6/8	1	1	6/8
1	2/8	–1/2	1/2	–2/8
2	–1/8			–1/8

Proceso de cuantificación y codificación

La cuantificación y codificación de las imágenes obtenidas como resultado de la transformada wavelet sigue una estructura compleja basada en una metodología conocida como EBCOT (*Embedded Block Coding with Optimized Truncation of the embedded bitstream*). Aunque los detalles son muy tediosos y el algoritmo tiene muchas fases de análisis y selección de píxeles compleja, la idea general es simple. El algoritmo de codificación selecciona los coeficientes más representativos de las imágenes transformadas y los cuantifica con un error de cuantificación prefijado. Estos coeficientes se transmiten siguiendo un orden decreciente en los niveles de descomposición. Posteriormente, se seleccionan más coeficientes significativos que se cuantifican con un error menor. Para los coeficientes transmitidos en la primera fase se transmiten bits adicionales para mejorar la precisión con la que se obtienen en el decodificador. El proceso se repite varias veces, transmitiendo cada vez los coeficientes con menor valor absoluto y refinando los coeficientes transmitidos en fases anteriores de manera que al final de la secuencia la imagen puede recuperarse con gran calidad o incluso sin pérdidas. La ventaja de esta estrategia es que el decodificador puede ir aproximando la imagen de forma progresiva a medida que está recibiendo mayor información en el *bitstream*. Además, otra particularidad es que la secuencia de bits puede interrumpirse en cualquier momento, ya sea porque se ha llegado a la precisión deseada en la reconstrucción o porque el número de bits totales excede un valor predeterminado (bpp fijados al iniciar el proceso de codificación).

Los símbolos se codifican mediante un codificador MQ (aritmético) que es parecido al que se utiliza en el estándar JPEG convencional. El codificador MQ también se utiliza en el estándar JBIG (transmisión de imágenes binarias – principalmente Fax).