PROYECTO REDES MULTIMEDIA

IMPLEMENTACIÓN DEL ESTÁNDAR DE COMPRESIÓN JPEG 2000

1. Sobre la descripción de las actividades y la coordinación de grupos

Las actividades consistirán en la realización obligatoria (en grupo) de trabajos teórico/prácticos relativos a la asignatura. Estos trabajos serán subproyectos de un proyecto general, que consiste en la implementación de un compresor de imagen JPEG 2000. Dichos trabajos serán tutorizados por el profesor, y expuestos en dos sesiones de teoría.

Los contenidos de dichos trabajos tendrán una parte teórica (investigación bibliográfica) y otra práctica (implementación, ejecución y evaluación de distintos componentes del compresor a desarrollar), y será necesaria la interacción entre los grupos durante el desarrollo de los mismos, y para la comprobación del funcionamiento adecuado de todo el compresor global.

Los grupos de actividades deberán tener tres componentes. Existe la posibilidad de formar grupos de dos componentes, bajo la condición de aceptar, si fuera necesario, a un posible tercer componente para terminar de cuadrar los grupos. En todo caso, no será considerado como mérito adicional el hecho de que un grupo esté formado por menos componentes, por lo que se aconseja que los grupos lo formen tres personas.

La composición de los grupos se hará apuntando a los componentes en la hoja que saldrá publicada en la puerta del laboratorio de prácticas de redes. En la composición de los grupos deberán detallarse el nombre y apellidos de los componentes encuadrándolos dentro de la actividad seleccionada.

Posteriormente, y una vez reservada una actividad en el tablón, se debe enviar un correo al profesor, indicando la composición del grupo, la parte del proyecto escogida, y el papel de cada uno de los integrantes del grupo.

En general, cada uno de los integrantes del grupo podrá asumir una de las siguientes responsabilidades de coordinación:

- Responsable de grupo, centrándose en la coordinación interna del grupo. Se encargará de planificar los objetivos según etapas, y de que estos plazos se cumplan en la medida de lo posible. También será el principal encargado de organizar las presentaciones de los subproyectos, tanto de la presentación en póster como de la presentación oral final del trabajo.
- Responsable de coordinación externa. Será la persona responsable de comunicación con los otros grupos, tanto para resolver dudas y necesidades comunes, como para realizar las pruebas finales y el montaje del compresor completo.
- Responsable de coordinación con el profesor. Su labor es resolver las dudas que tengan, tanto los componentes del grupo hacia el profesor, como el profesor hacia los alumnos, manteniendo en todo momento una realimentación para que

el profesor conozca la marcha de las actividades, y los principales problemas observados.

Las responsabilidades que acabamos de plantearos no son más que *roles* o papeles que os invitamos a interpretar. El objetivo es acercar el funcionamiento de las actividades al trabajo en grupo y la coordinación entre grupos que se exige en el mundo laboral. Sin embargo, ninguna de las responsabilidades que os hemos planteado está por encima de las otras, sino que simplemente apuntan a distintos cometidos.

2. Sobre la temática de los subproyectos

En este curso pretendemos realizar la implementación de un compresor de imagen basado en el estándar JPEG 2000, de forma similar a la implementación parcial del estándar JPEG que realizamos en prácticas.

En la Figura 1 se encuentra una descripción en forma de esquema del compresor/descompresor JPEG 2000 a implementar, destacando cada uno de los ocho subproyectos en los que se ha dividido el proyecto general. En concreto, la temática de cada uno de los subproyectos es la siguiente:

- Conversión RGB-YCbCr 4:2:0 e inversa. Esta primera etapa consiste en una conversión de planos de colores, de RGB a YCbCr. Puede ser realizada tanto con tipo de datos *floats*, para obtener unos valores exactos, como con enteros, para realizar una conversión reversible (sin posibles errores de redondeo).
- Transformada Wavelet (directa e inversa). Cada uno de los planos de luminancia y crominancia deben ser transformados del dominio inicial (espacial) al dominio de la transformada Wavelet, utilizando para ello un filtro paso bajo y otro paso alto, aplicado primero en filas y luego en columnas de forma sucesiva. La transformada puede ser implementada también siguiendo el esquema lifting, lo que facilita una implementación con enteros y sin pérdidas.
- Cuantización (y cuantización inversa). En este momento se hace un paso de coeficientes en coma flotantes a coeficiente usando una representación con enteros, utilizando para ello las operaciones definidas en el estándar.
- División en bloques de tamaño 64x64 (y reconstrucción a partir de bloques). A continuación, cada una de las subbandas resultantes de la transformada Wavelet se dividen en bloques de 64x64 coeficientes (o menos si fuera necesario).
- Codificación-decodificación siguiendo el algoritmo EBCOT. Cada uno de los bloques se codifican por planos de bits (realizando tres pasadas por cada plano de bits) con distintos contextos (es decir, distintas estadísticas en función de los coeficientes que rodean al coeficiente a calcular).
- Codificación (y decodificación) aritmética. El resultado de la codificación con EBCOT es una sucesión de símbolos binarios que se codifican con diferentes contextos. El codificador binario que comprime eficientemente esta sucesión en JPEG 2000 es el “MQ-encoder”.
- Formación del bitstream final. Partiendo del resultado del codificador EBCOT, y usando el MQ-coder, hay que ordenar el bitstream por planos de bits si se desea escalabilidad SNR, o por subbandas si se quiere escalabilidad espacial. Esta parte se encarga de realizar este tipo de reordenación. Además, como alternativa, se pueden aplicar algoritmos de optimización (como el de Lagrange) para seleccionar la parte óptima de cada bloque que se incorpora al bitstream final.

Otro aspecto que se puede tratar en este subproyecto es el formato del propio bitstream, que viene dado por el estándar JPEG 2000.

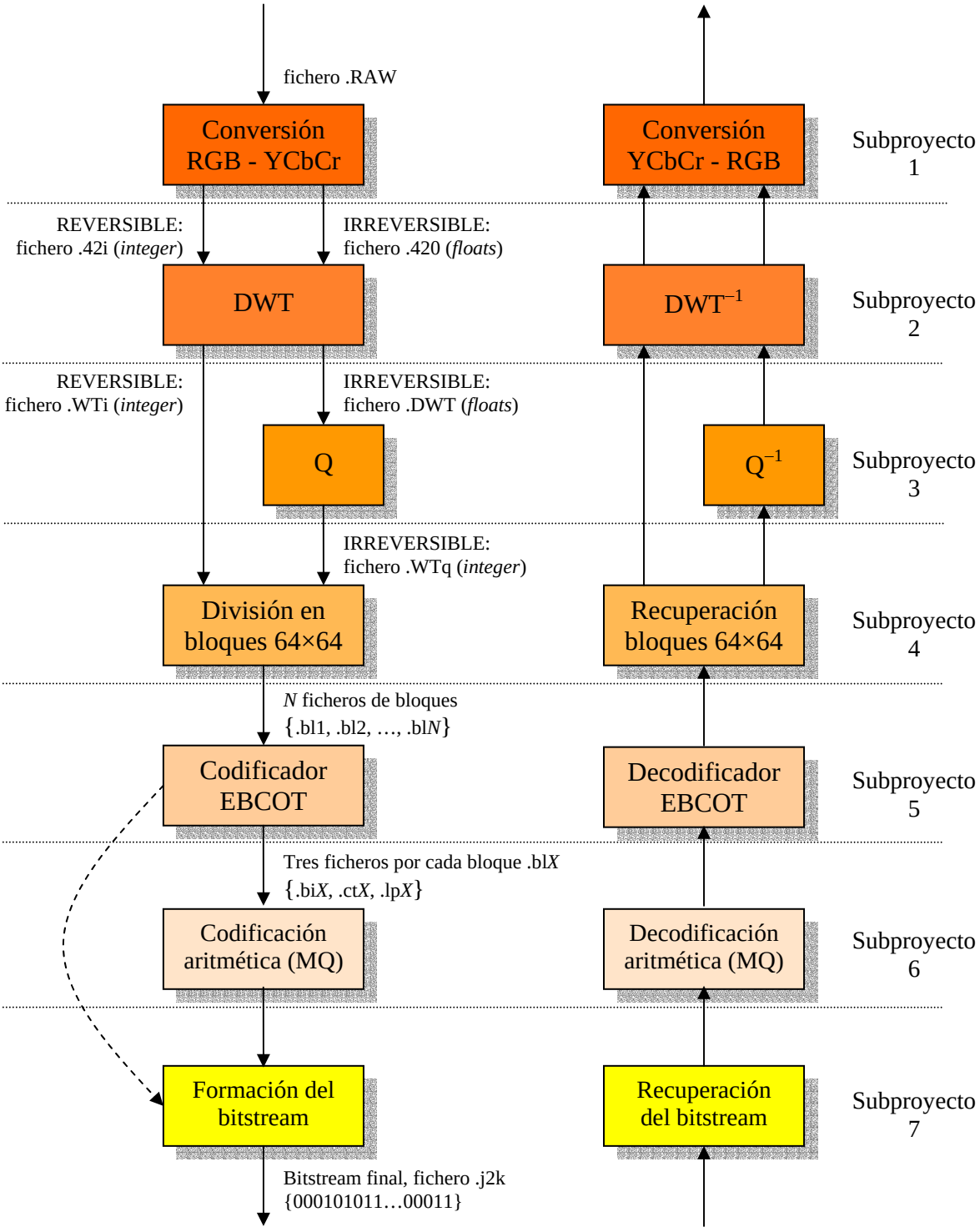


Figura 1: Esquema general del proyecto a implementar, dividido en subproyectos.

3. Sobre la implementación de los trabajos

Los grupos de un nivel se deberán coordinar con los del resto de niveles para la realización y prueba final de los trabajos. Los programas a desarrollar deben recibir los parámetros de configuración del usuario por medio de la línea de comandos, de forma similar a como se realiza en los programas de prácticas. Además, la forma de intercambiar documentación entre los distintos subproyectos es por medio de ficheros, recibiendo un fichero (o varios, según el nivel) del subproyecto del nivel anterior, y operando sobre estos datos para generar nuevos ficheros con el resultado del subproyecto, que se pasarían al siguiente nivel.

En el horario de prácticas, los componentes de los grupos dispondrán del laboratorio de redes y de la documentación facilitada para realizar las actividades de: (1) búsqueda de más documentación, (2) programación, y (3) realización de pruebas individuales y conjuntas. Así mismo, también dispondrán del profesor, al que se podrá consultar cualquier duda.

A continuación se describen algunos detalles sobre la implementación de los subproyectos. Aun así, se recuerda que es parte del trabajo la búsqueda de documentación relacionada con la implementación de cada subproyecto.

3.1 Conversión RGB-YCbCr

Este subproyecto inicia el proceso de la compresión, así que recibe la imagen a comprimir en formato .raw (en el que la imagen viene dada con los planos de colores RGB), y genera un fichero con la imagen recodificada a formato YCbCr con reducción de color 4:2:0.

Su implementación en coma flotante es similar a la que se realizó en la práctica de compresión, sin embargo, en este caso pedimos que el fichero resultante contenga los planos de luminancia (Y) y de crominancia (Cb y Cr) directamente almacenados en coma flotante (tipo *float*, de 4 bytes), y no en "*unsigned/signed char*" (o bytes). El fichero que se genera en este subproyecto, y que contendrá los planos de lumniancia y crominancia en coma flotante, debe tener una extensión ".420".

De forma opcional, se puede realizar esta misma conversión operando con tipo de datos enteros, de manera que el resultado sería un fichero que contuviera "*int*s" (entero con signo de 32 bits) en lugar de "*float*s". Este tipo de implementaciones permite garantizar la reversibilidad (ya que no hay errores de redondeo, como sí puede ocurrir al operar en coma flotante), pero son aproximaciones menos precisas y por lo tanto menos eficientes.

Para la implementación con enteros se deben utilizar las siguientes ecuaciones:

$$Y = \left\lfloor \frac{R + 2G + B}{4} \right\rfloor \qquad U = R - G \qquad V = B - G$$

El valor original puede ser reconstruido sin pérdidas como:

$$G = Y - \left\lfloor \frac{U+V}{4} \right\rfloor \qquad R = U + G \qquad B = V + G$$

Además, para ser reversible, no se puede realizar ninguna reducción de color. El fichero generado por esta vía debe de tener la extensión ".44i"

k	Análisis paso bajo, $\{h_k\}$	k	Análisis paso alto, $\{g_k\}$
0	+ 0.602949018236360	0	+ 0.557543526228500
± 1	+ 0.266864118442875	± 1	– 0.295635881557125
± 2	– 0.078223266528990	± 2	– 0.028771763114250
± 3	– 0.016864118442875	± 3	+ 0.045635881557125
± 4	+ 0.026748757410810		

k	Síntesis paso bajo, $\{\tilde{h}_k\}$	k	Síntesis paso alto, $\{\tilde{g}_k\}$
0	+ 0.557543526228500	0	+ 0.602949018236360
± 1	+ 0.295635881557125	± 1	– 0.266864118442875
± 2	– 0.028771763114250	± 2	– 0.078223266528990
± 3	– 0.045635881557125	± 3	+ 0.016864118442875
		± 4	+ 0.026748757410810

Tabla 1: Filtro bi-ortogonal 9/7 para implementar la DWT. En JPEG2000 este filtro se suele aplicar con una normalización (1,2) (se multiplican por dos las componentes del filtro análisis paso alto y síntesis paso bajo), de manera que el banco de filtros estará formado por $\{h_k\}$, $\{2g_k\}$, $\{2\tilde{h}_k\}$, $\{\tilde{g}_k\}$

3.2 DWT/IDWT (Discrete Wavelet Transform)

A continuación se debe recibir los tres planos de luminancia y crominancia en coma flotante (fichero ".420") y generar la transformada Wavelet de cada uno de los planos.

La transformada Wavelet a utilizar es la que por defecto usa JPEG 2000, que está formada por el filtro paso-bajo de análisis $\{h_k\}$ y el filtro paso alto de análisis $\{g_k\}$ descritos en la tabla 1. Para implementar la transformada 2D, en primer lugar debes implementar la transformada 1D, y después aplicar la versión 1D a la imagen, primero por filas y luego por columnas. **Antes de pasar a implementar la versión 2D, se recomienda implantar en primer lugar la transformada 1D directa en inversa, y probar su correcto funcionamiento mediante la transformación directa e inversa de un vector.**

Un problema a resolver en la transformada es cómo operar con los valores próximos a los bordes, en los que no se dispone de suficientes muestras para aplicar al filtro (por ejemplo, para aplicar un filtro a la primera muestra de la fila no disponemos de valores a la izquierda de las muestras). Para poder aplicar el filtro en estos casos hay distintas alternativas, pero la que se propone es usar extensión simétrica (es decir, si el vector inicial de muestras es "abcde…", replicamos el borde simétricamente con las muestras

del propio vector de la siguiente forma: “edcbabcde…”, esta solución también se puede adoptar en el extremo final del vector).

Después de aplicar ambos filtros, el resultado se debe reducir, haciendo un “subsampling” entre dos, de forma que sólo nos quedamos con una de cada dos muestras obtenidas (en las muestras de bajas frecuencias nos quedaremos con aquellas que están en posiciones pares, considerando 0 la primera posición, y en el resultado del filtrado paso alto nos quedamos con los coeficientes resultantes en posiciones impares). El resultado de la descomposición se situará en el propio vector de entrada, de manera que la mitad inferior tendrá el resultado de aplicar el filtro paso-bajo, y la otra mitad el resultado de aplicar el filtro paso-alto. De esta forma, habremos calculado un primer nivel de descomposición. Para obtener niveles adicionales, se puede repetir la misma operación, de manera sucesiva, sobre los coeficientes resultantes de aplicar el filtro de baja frecuencia.

Para calcular un nivel de transformada inversa, simplemente hay que aplicar un “upsampling” para retornar cada una de las señales a la resolución inicial, incorporando “0” entre las muestras (que sustituirán a los coeficientes que se descartaron en el “subsampling”), y aplicando los filtros de síntesis paso-bajo y síntesis paso-alto de la tabla 1 sobre cada una de las subbandas restauradas al tamaño original (no te olvides de aplicar extensión simétrica aquí también). Por último hay que sumar el resultado del proceso del filtrado, y habremos recuperado la señal original.

El proceso de descomposición y reconstrucción Wavelet 1D que acabamos de ver viene descrito de forma esquemática (para un único nivel de descomposición) en la figura 2.

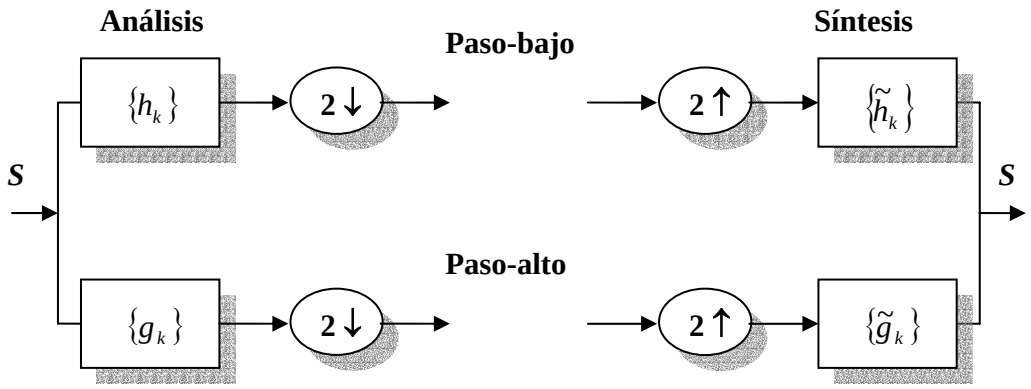


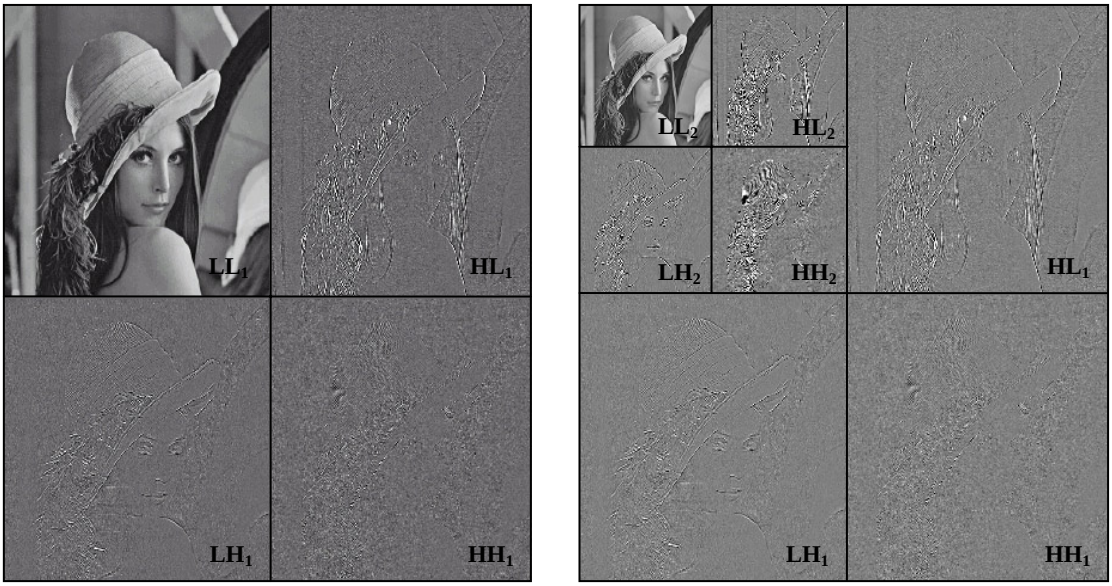
Figura 2: Un nivel de descomposición wavelet de una señal, y su reconstrucción obteniendo la señal original.

Un ejemplo de su extensión a 2D viene dado en la figura 3. En la figura 3(a) podemos observar el resultado de aplicar un nivel de descomposición Wavelet, por filas y columnas tal y como se ha descrito anteriormente. Además, en esta figura podemos ver la nomenclatura que se utiliza para denominar a cada una de las subbandas resultantes: HL para la subbanda que contiene información de baja frecuencia horizontal y alta frecuencia vertical, LH la que contiene justamente lo contrario, alta frecuencia horizontal y baja frecuencia vertical, HH que contiene alta frecuencia tanto en horizontal como en vertical, y la LL, que contiene bajas frecuencias en ambos casos. Además, se indica con un subíndice el nivel de descomposición al que pertenece cada

subbanda, como se puede ver en la figura 3(b), a la que se la ha aplicado dos niveles de descomposición.

El resultado de aplicar la transformada Wavelet a cada uno de los planos YCbCr se debe guardar en otro fichero de números en coma flotantes, cuya extensión es ".DWT".

Por último, y de manera opcional, se puede investigar cómo implementar la transformada Wavelet con enteros siguiendo el esquema lifting, en el que la señal se descompone usando pasos sucesivos de predicción y actualización con la transformada Wavelet biortogonal 5/3. El resultado, en este caso, debería quedar en un fichero de enteros con extensión ".DWi".



(a) (b)

Figura 3: (a) Un nivel de descomposición wavelet de la imagen de entrada, (b) dos niveles de descomposición wavelet, en el que la subbanda LL_1 de (a) ha sido descompuesta.

3.3 Cuantización(Q y Q^{-1})

El siguiente paso consiste en tomar el fichero de flotantes (.DWT) y convertirlo a enteros realizando una cuantización de su contenido, y generando así un fichero de extensión (.WTq). Las fórmulas para realizar la cuantización directa e inversa se pueden encontrar en diversos artículos técnicos y en el propio estándar.

3.4 División (reconstrucción) en bloques de 64x64

Esta parte del proyecto debe tomar cada plano de la imagen y dividirlo en bloques de 64x64 componentes. Algunos de los bloques pueden ser menores, bien porque la propia subbanda a tratar sea menor, o bien porque es uno de los bloques que se sitúan en los límites de la subbanda, y por tanto contengan los últimos coeficientes sobrantes (siempre que la subbanda no tenga dimensiones múltiplos de 64).

Cada uno de los bloques se codificará por separado, en un fichero independiente de 64x64 enteros. Como las dimensiones del bloque podrían ser distintas de 64x64, es

necesario generar una pequeña cabecera indicando la dimensión exacta de cada bloque (ancho x alto). Para su posterior tratamiento, también es necesario indicar la subbanda a la que pertenece dicho bloque (tanto el tipo: HH-HL-LH ó LL, como el nivel: 1, 2, 3, 4…). Esta información también aparecerá en la cabecera de la función.

Finalmente, el fichero que contenga el primer bloque generado tendrá como extensión ".bl1", el segundo ".bl2", …, el *n*-ésimo extensión ".bl*n*". Además, en el decodificador los bloques se deben reconstruir en el mismo orden y misma posición en la que se generaron.

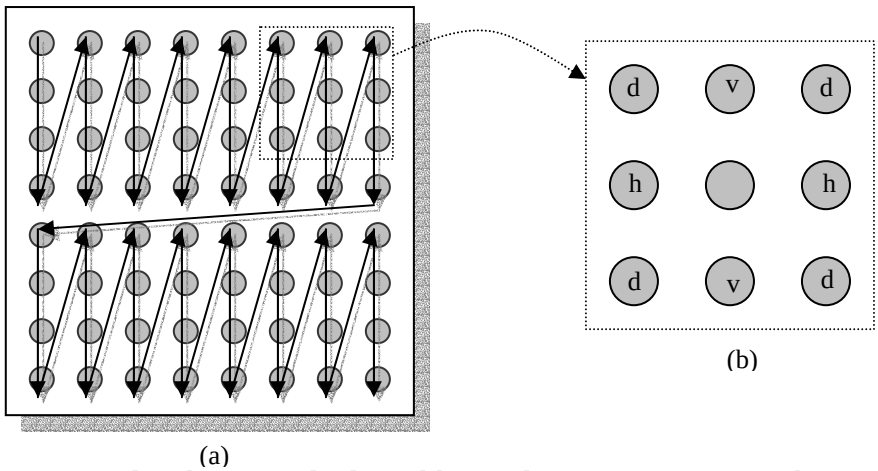


Figura 4: (a) Orden de recorrido de un bloque de 8x8 componentes de JPEG 2000, y (b) contexto empleado para codificar los coeficientes, formado por sus ocho vecinos (dos horizontales, dos verticales y cuatro diagonales).

3.5 Codificador/decodificador EBCOT

Éste es realmente el corazón de JPEG 2000. Realiza una codificación completa de cada bloque generado en el subproyecto anterior, recorriendo este bloque por planos de bits, con tres iteraciones (o pasadas) por cada plano. De esta manera, **para cada bloque se generará un bitstream independiente**.

Como ya hemos visto, en esta fase cada bloque se codifica por planos de bits, empezando por el más significativo. Para entender este codificador, en primer lugar vamos a definir la significancia de un coeficiente respecto a un plano de bit determinado. Decimos que un coeficiente *c* es significativo respecto al plano de bit *b*, si necesita más de *b* bits para ser codificado. Por ejemplo, un coeficiente de valor hexadecimal 0x5A = 1011010b, no es significativo respecto al plano de bit 9, pero sí lo es respecto a los planos de bit 6, 5, 4, …, 0. Además, consideramos que sus bits significativos son todos aquellos que están por debajo de su primer plano de bit significativo (en este caso, 011010b).

La codificación EBCOT se puede resumir como sigue: en primer lugar determinamos cuál es el mayor de los coeficientes en el bloque, y una vez identificado este coeficiente se calcula cuántos bits son necesarios para codificarlo (en valor absoluto). Por ejemplo, si el mayor coeficiente del bloque es -61 (en decimal), sabemos que su representación binaria (sin signo) es 111101b, por lo que necesita 6 bits para ser codificado. Así, en la codificación por planos de bit realizada por EBCOT, codificaremos en primer lugar el

sexto bit de todos los elementos del bloque, después codificaremos el quinto bit, después el cuarto, hasta codificar el primer bit de todos los elementos del bloque.

Sin embargo, cada plano de bit no se codifica de una sola vez, sino que se realizan varias pasadas, intentando identificar qué coeficientes se vuelven significativos en este plano de bit (por ejemplo, 0x5A se vuelve significativo en el plano 6, ya que en el 7 no lo era). De esta forma sabremos cuántos bits necesita cada coeficiente (en el ejemplo anterior sabremos que 0x5A necesita 6 bits, ya que en el plano de bit 6 pasa a ser significativo).

En concreto, cuando vamos a codificar un plano de bit *n*, podemos encontrarnos con dos tipos de coeficientes respecto a ese plano, (1) aquellos coeficientes que son significativos respecto a este plano de bit (es decir, ocupan más de *n* bits) y (2) aquellos que aun no son significativos (ocupan *n* bits o menos). De las tres pasadas que se realizan en cada plano de bits, dos se centran en codificar el *n*-ésimo bit de aquellos coeficientes que aun no son significativos (estas pasadas son las de propagación y de limpieza), y la tercera se usa para codificar el bit *n*-ésimos de aquellos coeficientes que ya son significativos (porque se hicieron significativos en capas de bit anteriores).

Para aumentar la eficiencia en la compresión, se usarán distintos contextos para codificar los bits usando el codificador aritmético con distintas probabilidades estadísticas. Para entender mejor la necesidad de uso de contextos es interesante hacer una pequeña explicación de cómo funciona la codificación aritmética en JPEG 2000. Sabemos que un codificador basado en la entropía (como Huffman o codificación aritmética) necesita conocer las probabilidades estadísticas de los distintos símbolos para ser eficientes (estas probabilidades se obtienen muchas veces a partir del uso de histogramas, que nos indican los símbolos más frecuentes). Al empezar a codificar una imagen, a priori se desconoce cuales son los símbolos que van a aparecer con más frecuencia, sin embargo, conforme se van codificando los símbolos, podemos ir rellenando de forma dinámica un histograma usando los propios valores que se codifican. A este tipo de codificación se le llama codificación adaptativa. Por otro lado, sabemos que un codificador basado en la entropía será más eficiente si concentramos probabilidades. Los contextos nos sirven para definir distintos modelos estadísticos (es decir, histogramas) según las condiciones de los símbolos que vamos a codificar, de forma que usaremos el mismo histograma (dicho de otra forma, el mismo contexto) para codificar símbolos que prevemos que van a ser similares. Así concentramos probabilidades.

Por ejemplo, si quiero codificar el bit *n*-ésimo de un coeficiente que hasta ahora no es significativo, y sé que casi todos sus vecinos no son significativos, lo más normal este bit sea un 0 (es decir, que siga siendo no significativo), sin embargo, si sé que sus vecinos son significativos, posiblemente este coeficiente pase a ser significativo en este plano de bit (es decir, sea un 1). Por tanto, puedo definir dos contextos distintos, y usaré uno u otro según la significancia de sus vecinos, de manera que concentraré la mayor parte de bits a 1 en un histograma (o contexto), y la mayor parte de bits a 0 en el otro. De esta forma la compresión será mayor. Fíjate que la condición que hemos empleado en este caso para distinguir el contexto a usar es la significancia de los vecinos, y nos ha permitido distinguir dos contextos, pero podemos emplear muchas otras condiciones para definir muchos otros contextos.

Veamos a continuación con más detalle las tres pasadas que se realizan por plano de bit.

- En la primera pasada que se efectúa, llamada paso de propagación (o *propagation pass*), se codifica la significancia de aquellos coeficientes que no fueron significativos en planos de bits anteriores, pero que probablemente pasarán a ser significativos en este plano de bit. En concreto, se considera que un coeficiente es probable que pase a ser significativo si al inicio de este paso tiene al menos un vecino que ya es significativo. Para codificar la significancia de cada uno de los coeficientes que cumplen con esta condición indicamos con un "1" si el coeficiente realmente pasa a ser significativo o con un "0" si permanece no significativo (es importante que entiendas que el hecho de que pensemos que el coeficiente va a pasar a ser significativo no quiere decir que finalmente esto ocurra). En este paso usaremos nueve contextos distintos en función de la significancia de sus ocho coeficientes vecinos (ver vecinos en figura 4). Además, para cada coeficiente que pase a ser significativo, tenemos que codificar también su signo, utilizando para ello un codificador aritmético con cinco contextos. La asignación de contextos exacta debe consultarse en el estándar o en artículos relacionados.
- En la segunda pasada, llamada paso de refinamiento de la magnitud (o *magnitude refinement pass*), se codifica un bit por cada coeficiente que ya fue localizado como significativo en otros planos de bits anteriores. En este caso, si estamos codificando el plano de bit n-ésimo, simplemente debemos de codificar el bit n-ésismo de los coeficientes ya significativos.
- En la tercera pasada, codificamos la significancia del resto de los coeficientes (es decir, aquellos que no fueron significativos en los planos de bit previos y que probablemente permanecerán no significativos en este plano de bit). Esta última pasada de cada plano de bit se denomina pasada de limpieza (o *clean up pass*), y para ella se usan los mismos contextos que en el paso de propagación. Sin embargo, la principal novedad de este paso es que incluye un modo tipo "run-length" para reducir complejidad cuando aparecen varios ceros seguidos (esto es, varios coeficientes no significativos seguidos). Aunque la implementación del modo run-length es obligatoria en el estándar, en nuestras actividades la dejamos como opcional.

El orden en el que se debe de recorrer cada bloque es en columnas de cuatro coeficientes, tal y como se indica en la figura 4.

Fíjate que el resultado de todos los pasos es siempre codificar bits con diferentes contextos. Ésta es la información que le entrará al codificador aritmético (MQ-coder), sin embargo, nosotros debemos generar esta información directamente sobre un fichero, para poder desacoplar la implementación de ambos subproyectos.

Por tanto, por cada bloque a codificar (bloque n-ésimo, con extensión .bl*n*), se debe generar un fichero de bits, que contenga la información binaria codificada (fichero .bi*n*) y otro que indique el contexto a usar por cada bit (fichero .ct*n*). Además, para su posterior reordenación, también es necesario incluir un fichero que indique la longitud de cada una de las pasadas (iteraciones o pasos) que se han efectuado, guardando el número de bits codificados en cada pasada (usaremos un fichero de extensión .lp*n*). Este fichero también debe contener información sobre el primer plano de bit codificado (que será el primero que contiene algún coeficiente significativo).

3.6 Codificador/decodificador binario aritmético (MQ-coder)

JPEG 2000 utiliza un codificador aritmético binario (es decir, cuyo alfabeto de entrada es {0, 1}) con 18 contextos (es decir, 18 modelos estadísticos o "histogramas" distintos). Este codificador especial se denomina "MQ-encoder", y es una versión de menor complejidad que el codificador aritmético habitual. En este subproyecto se debe analizar el funcionamiento de un codificador aritmético estándar, y posteriormente centrarse en el MQ-coder.

Dentro del estándar JPEG 2000 se proporciona un diagrama de flujo en el que se describe de forma detallada cómo implementar este codificador. En este caso, este subproyecto se puede basar en alguna versión del "MQ-encoder" ya implementada en el software de referencia de JPEG 2000 (por ejemplo, Jasper), para hacer una evaluación detallada de su uso y una descripción del mismo.

3.7 Formación del bitstream

En este subproyecto se parte de los ficheros con la información sobre los bits codificados por EBCOT (tanto los propios bits como los contextos) y, utilizando una de las versiones disponibles del codificador MQ-encoder (por ejemplo la incluida en Jasper) que no tiene el por qué ser analizada (esto es tarea del subproyecto 6), hay que generar un bitstream final, que reordene los coeficientes según la escalabilidad deseada (escalabilidad SNR o espacial).

De manera opcional, se puede construir el bitstream según el formato oficial de JPEG 2000, o al menos realizar un estudio de cómo es la estructura de este formato de fichero.

También como ampliación se podría reordenar el bitstream aplicando algoritmos de optimización (como el de Lagrange) para seleccionar la parte óptima de cada bloque que se incorpore al bitstream final.

4. Sobre la tarea a realizar

Lo primero que hay que hacer para realizar el proyecto es formar grupos de tres personas y asignar responsabilidades de coordinación (interna, externa y con el profesor). Una ver realizado esto, se deberá seleccionar tema en función del trabajo preferido por los miembros del grupo, para a continuación apuntarse en la hoja que a tales efectos se encuentra disponible. Es importante destacar que sólo se podrá asignar tres subproyectos del mismo tipo, por lo que dicha asignación se hará en riguroso orden de inscripción, de manera que si el trabajo que el grupo había seleccionado ya ha sido escogido por otros tres grupos, tendrá que seleccionarse un nuevo trabajo. Además hay que destacar que los subproyectos 1, 3 y 4 se deben realizar de manera conjunta, es decir, no se pueden asignar individualmente.

La primera tarea a realizar, una vez decidido el subproyecto, es documentarse sobre cómo implementar el proyecto. Para esto se pueden consultar revistas técnicas relacionadas o el propio estándar (disponible bajo solicitud al profesor). Es importante realizar esta tarea pronto, porque dentro de pocas semanas se hará una presentación de la teoría asociada a cada subproyecto por medio de la modalidad de "póster".

Una vez documentado suficientemente el trabajo, empezará la fase de implementación. Para ello se dispondrá de una unidad básica para la lectura/escritura de los ficheros descritos en el punto anterior que, en caso de ser necesario, podrá ser modificada por parte de los propios grupos. También se facilitará un fichero de entrada por subproyecto, para poder disponer de información que permita que todos los grupos empiecen a trabajar en paralelo, sin tener que depender directamente del grupo anterior.

Posteriormente, se deberá realizar una memoria sobre el trabajo efectuado por cada grupo y los resultados obtenidos, que además será presentada de forma oral en las últimas sesiones de teoría.

Por último, se podrá realizar una prueba conjunta del sistema, para la que al menos los coordinadores externos deberán reunirse en la última de las sesiones de prácticas para ejecutar los programas secuencialmente, de forma que se pueda evaluar el resultado del sistema global. Los coordinadores externos podrán presentar también los resultados de estas pruebas en la sesión de presentación de trabajos, si éstos son favorables. También se pueden incluir estos resultados en la memoria de prácticas de cada grupo.