ANEXO B. ESTANDAR JPEG2000 PARA COMPRESION DE IMAGENES

B.1 Especificaciones del estándar JPEG2000:

El estándar JPEG2000 soporta compresiones con o sin pérdidas de imágenes de uno o múltiples componentes (escalas de grises o colores). Adicionalmente a esta funcionalidad de compresión básica se han generado otros servicios a proveer, incluyendo: 1) recuperación progresiva de una imagen por fidelidad o resolución; 2) codificación de áreas de interés (*Region of interest – ROI)* en las que diferentes zonas de la imagen pueden ser compresas a diferentes rangos de fidelidad; 3) acceso aleatorio a regiones particulares de una imagen sin tener que decodificar el archivo completo; 4) un formato de archivos flexible que permite agregar información de opacidad y secuencia en una imagen y 5) una buena resistencia a los errores. Debido a su funcionamiento excelente para codificación de imágenes y sus servicios adicionales, JPEG2000 posee una base de aplicaciones muy amplia. Unos de sus posibles usos incluyen: almacenamiento de imágenes, Internet, exploración de contenido web, fotografía digital, toma de imágenes para uso médico, publicación de imágenes, entre otros.

B.2. Por qué JPEG2000?

Los trabajos en el estándar JPEG2000 empezaron desde una llamada a contribuciones en Marzo del 1997 [20]. El propósito de implementar un nuevo estándar se basaba en dos argumentos. Primero, tendría que arreglar un buen número de carencias del estándar JPEG. Segundo, el estándar debía proveer a los usuarios de más servicios que el estándar original. Tales puntos definieron varios objetivos para el nuevo estándar, el cual debía: 1) permitir una compresión eficiente con o sin pérdidas dentro de un sistema único de codificación, 2)

proveer una calidad superior en las imágenes, tanto objetiva como subjetivamente, en rangos altos de compresión, 3) permitir características adicionales como escalabilidad de resolución, codificación de ROI, y un formato de archivo más flexible, 4) evitar una carga computacional excesiva. Sin duda, gran parte del éxito del estándar JPEG original es debido a su característica libre. En consecuencia, se han hecho esfuerzos considerables para asegurar que existe un modelo del códec JPEG2000 básico, libre de costos.

Tabla I: Partes del Estándar.

Parte	Título	Contenido	Documento
1	*Core coding system*	Especifica la base (de funcionalidad mínima) del códec JPEG2000	14
2	*Extensions*	Especifica las funcionalidades adicionales que son útiles en algunas aplicaciones pero no son soportadas por todos los códec.	18
3	*Motion JPEG 2000*	Especifica las extensiones de JPEG 2000 para la compresión de video.	21
4	*Conformance testing*	Especifica el procedimiento a usarse para verificación del estándar.	22
5	*Reference software*	Provee aplicaciones guía para uso libre de los implementadores.	23
6	*Compound image file format*	Define el formato de archivo usado para documentos compuestos.	24
8	*Secure JPEG 2000*	Define mecanismos de acceso condicional, integridad/autentificación y protección de propiedad intelectual.	*
9	*Interactivity tools, APIs and protocols*	Especifica un protocolo cliente/servidor para transportar los datos de las imágenes a través de redes de datos.	*
10	*3D and floating-point data*	Provee las extensions para manejo de datos volumétricos (3D) y datos de punto flotante.	*
11	*Wireless*	Provee herramientas de codificación de canal y protección de errores en comunicaciones inalámbricas.	*
12	*ISO base media file format*	Define un formato de archivo común entre JPEG2000 y MPEG-4.	25
13	*Entry-level JPEG 2000 encoder*	Define un codificador JPEG2000 a nivel de entrada.	*

* Estos aspectos están aún en desarrollo.

B.3. Estructura del estándar

El estándar JPEG2000 está dividido en numerosas partes, con los segmentos en la Tabla I siendo definidos aun. Por conveniencia, se referirá al codificador definido en la parte primera del estándar [14] como el codificador de base. El codificador de base es simplemente el sistema de mínima funcionalidad del codificador JPEG2000. La segunda parte [18] del estándar define ciertas extensiones al codificador de base que son útiles para ciertas aplicaciones, mientras que la tercera parte [21] define las extensiones para la compresión de video. La parte 5 [23] presenta dos implementaciones de software del codificador de base, y la parte 4 [22] muestra una metodología de testeo de aplicaciones de acuerdo con el estándar. Por ahora nos limitaremos a la definición del codificador de base. Algunas de las extensiones incluidas en la parte 2 serán discutidas también. A menos que se indique lo contrario, esta explicación cubre exclusivamente al sistema básico.

En su mayor parte, el estándar JPEG2000 es diseñado desde el punto de vista del decodificador. Entiéndase, el decodificador está definido muy precisamente con muchos detalles normativos, mientras que muchas partes del decodificador están especificados menos rígidamente. Los diseñadores deben tener muy claro el diferenciar las cláusulas normativas de las informativas en este estándar. Para el propósito de este trabajo, sin embargo, se harán estas distinciones cuando sean absolutamente necesarias.

B.4. Codificador JPEG2000

Al introducir el estándar JPEG2000, estamos en la capacidad de examinar el codificador JPEG2000 en detalle. Este códec se basa en técnicas de codificación de wavelet/subbanda [26,27]. Maneja compresiones con y sin pérdidas usando el mismo sistema de transformadas, y toma las ideas del sistema de codificación de bloque anidada con truncado optimizado (*Embedded block coding with optimized truncation – EBCOT*) [28-30]. Para facilitar la codificación con y sin pérdidas de una manera eficiente, se han definido dos transformadas, una transformada reversible de entero a entero [31-33], y una transformada irreversible de entero a real. Para transformar los datos a código, se usan técnicas de

codificación de planos de bits [34]. Para los codificadores de entropía, se ha elegido un codificador aritmético binario adaptativo basado en el contexto – más específicamente el codificador MQ del estándar JBIG2 [35]. Dos niveles de sintaxis han sido empleados para representar la imagen codificada: una serie de datos y una sintaxis del formato en el archivo. La sintaxis del bloque de datos es similar a la usada en el estándar JPEG.

Esta sección estará estructurada en donde se discute el modelo de la imagen en las secciones A al C. en la sección D examina la estructura básica del códec. De las secciones E a la M se explica detalladamente cada segmento. De allí, las secciones N y O explican la sintaxis que se usan para representar una imagen codificada. Por último, la sección P describe algunas de las extensiones incluidas en la parte 2 del estándar.

B.4.1. Modelo de la imagen.

Antes de examinar el funcionamiento interno del códec, es importante entender el modelo de imagen que emplea. Desde el punto de vista del códec, una imagen está compuesta de uno o más componentes (hasta un límite de 2^{14}), como se muestra en la *figura 1*(a). Como se muestra en la *figura 1*(b), cada componente consiste de un arreglo rectangular de muestras. Los valores de muestreo en cada componente son valores enteros, y pueden tener o no signo con una precisión desde 1 hasta 38 bits por muestra. El signo y la precisión de los datos son especificados por separado en cada componente.

Todos los componentes están asociados a una extensión espacial común en la imagen, pero representan una información espectral o auxiliar diferente. Por ejemplo, una imagen de color RGB tiene tres componentes, cada componente representando los colores de plano en rojo, verde y azul. En el caso de las imágenes en escalas de grises, sólo hay un componente, correspondiente al plano de la luminancia. Los componentes de una imagen no necesitan ser muestreados a una misma resolución. Consecuentemente cada componente puede tener un distinto tamaño. Por ejemplo, cuando las imágenes a color están representadas es un espacio de color/transparencia, la información de las transparencias es muestreada a menor grado que la información del color.

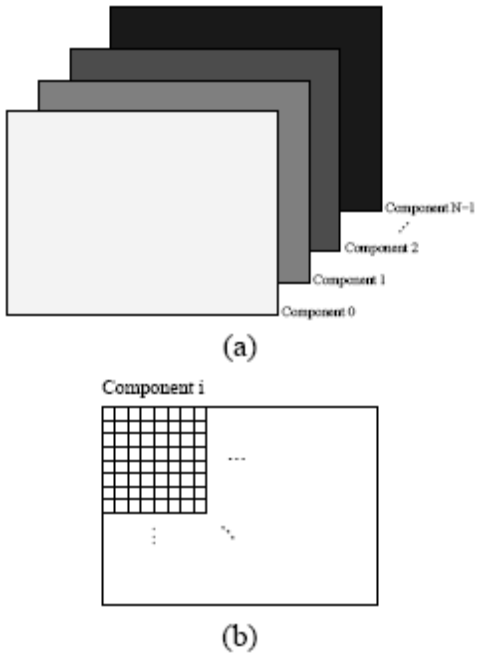

Figura 1. Modelo de una imagen. (a) *Una imagen con N componentes.* (b) *Componentes individuales.*

B.4.2. Malla de Referencia

Dada una imagen, el códec describe la geometría de los múltiples componentes con una malla rectangular llamada malla de referencia. La malla de referencia tiene una forma general mostrada en la figura 2. La malla tiene un tamaño de $Xsiz \times Ysiz$ con el origen localizado en la esquina superior izquierda. La región con su esquina superior izquierda en $(XOsiz, YOsiz)$ y su esquina inferior derecha en $(Xsiz - 1, Ysiz - 1)$ es conocida como el área de imagen, y corresponde a los datos de imagen a ser representados. El ancho y alto de la malla de referencia no puede exceder $2^{32} - 1$ unidades, imponiendo un límite a la imagen que puede ser manejada por el códec.

Todos los componentes son mapeados en el área de imagen de la malla de referencia. Como los componentes pueden no estar muestreados a la misma resolución de la malla de referencia, se requiere de información adicional para establecer este mapeo. Por cada componente, se debe indicar el periodo de muestreo horizontal y vertical en unidades de la

malla de referencia, notados como $XRsiz$ y $YRsiz$, respectivamente. Estos dos parámetros especifican una malla (rectangular) de muestreo cuyos puntos horizontales y verticales son múltiplos de $XRsiz$ y $YRsiz$, respectivamente. Todos los puntos que pertenecen al área de imagen, constituyen las muestras del componente en cuestión. Entonces, en términos del sistema de coordenadas, un componente tendrá un tamaño de $\left(\frac{Xsiz-XOsiz}{XRsiz}\right) \times \left(\frac{Ysiz-YOsiz}{YRsiz}\right)$ con su esquina superior izquierda en $\left(\frac{XOsiz}{XRsiz}, \frac{YOsiz}{YRsiz}\right)$. Hay que notar que la malla de referencia impone una alineación particular entre las muestras entre los varios componentes.

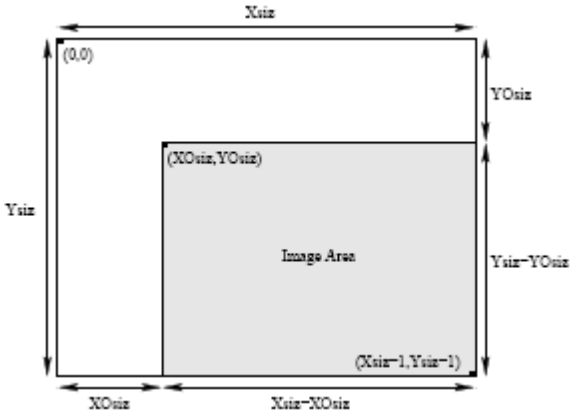

Figura 2. Malla de Referencia.

Basados en eso, el tamaño del área de imagen es $(Xsiz - XOsiz) \times (Ysiz - YOsiz)$. De una imagen dada, se pueden elegir muchas combinaciones de $Xsiz, Ysiz, XOsiz, YOsiz$ para obtener una misma imagen con unos mismos tamaños. Uno se puede preguntar porque no se asignan los parámetros $XOsiz\ y\ YOsiz$ en cero y que $Xsiz\ y\ Ysiz$ sean el tamaño de la imagen. Resulta que hay varias ventajas a los parámetros de $XOsiz\ y\ YOsiz$ (manteniendo el área de la imagen constante). Tales cambios afectan el comportamiento del codificador, como se describirá luego. Este comportamiento permite, entre otras cosas, el que se haga más eficientemente ciertas operaciones en las imágenes codificadas, como recortar segmentos, voltear horizontal y verticalmente una imagen, o rotarla por un múltiplo de 90 grados.

B.4.3. Teselación.

En algunos casos, una imagen puede ser muy grande en comparación con la cantidad de memoria permitida en el códec. En consecuencia, no siempre es útil el codificar la imagen íntegra como una unidad única. Para resolver este problema, el codificador permite a una imagen ser dividida en piezas más pequeñas, cada una de ellas codificada por separado. Más específicamente, una imagen es particionada en una o más regiones rectangulares llamadas teselas. Como se muestra en la figura 3, esta partición se genera con respecto a la malla de referencia al superponer la malla de referencia con una malla de teselación de espaciamientos $XTsiz$ y $YTsiz$, respectivamente. El punto origen de la malla de teselación se alinea con el punto $(XTOsiz, YTOsiz)$. Las teselas tienen un tamaño nominal de $XTsiz \times YTsiz$, pero las que están en los bordes de la imagen pueden tener un tamaño que difiere del tamaño nominal. Las teselas son numeradas en orden incremental (empezando por cero).

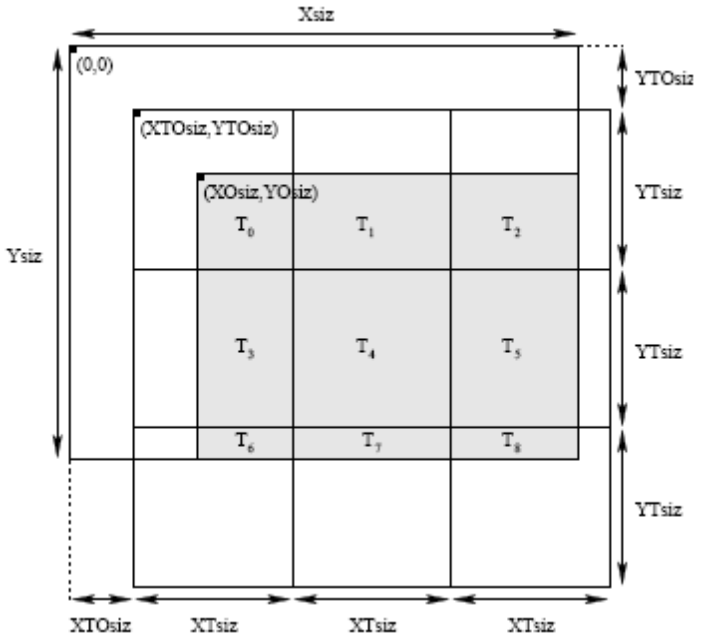

Figura 3. Teselación en la malla de referencia.

Al mapear la posición de cada tesela desde la malla de referencia a los sistemas de coordenadas de los componentes individuales, habrá una partición de los componentes en sí. Por ejemplo, supongamos que una tesela tiene su esquina superior izquierda e inferior derecha en las coordenadas (tx_0, ty_0) y $(tx_1 - 1, ty_1 - 1)$, respectivamente. Entonces en el espacio de coordenadas de un componente, esta tesela tendrá sus límites en las coordenadas (tcx_0, tcy_0) y $(tcx_1 - 1, tcy_1 - 1)$, respectivamente, donde:

$$(tcx_0, tcy_0) = ([tx_0/XRsiz], [ty_0/YRsiz]) \qquad (1a)$$

$$(tcx_1, tcy_1) = ([tx_1/XRsiz], [ty_1/YRsiz]) \qquad (1b)$$

Estas ecuaciones corresponden a la ilustración en la figura 4. La porción de un componente que corresponde a una tesela es referido como componente de tesela. Aunque la malla de teselación es regular con respecto a la malla de referencia, es importante notar que la malla no es necesariamente regular con respecto a los sistemas de coordenadas de los componentes.

B.4.4. Estructura del Códec.

La estructura general del códec es mostrada en la figura 5 con el diagrama de bloques del codificador en la figura 5(a) y el decodificador en la figura 5(b). Desde estos diagramas podemos definir cada uno de los procesos asociados al códec: 1) pre-procesamiento/post-procesamiento, 2) transformada intercomponente, 3) transformada intracomponente, 4) cuantización/decuantización, 5) codificación de primer orden, 6) decodificación de segundo orden, y 7) control de la tasa de compresión. La estructura del decodificador imita esencialmente a la del codificador. Entiéndase, con la excepción del control de la tasa de compresión, hay una correspondencia de uno a uno entre el codificador y el decodificador. Cada bloque funcional en el decodificador invierte exacta o aproximadamente el efecto de su bloque correspondiente en el codificador. Como las teselas son codificadas por separado, la imagen de entrada es (al menos en concepto) procesada una tesela a la vez. En la siguiente sección, cada uno de los procesos se examina con más detalle.

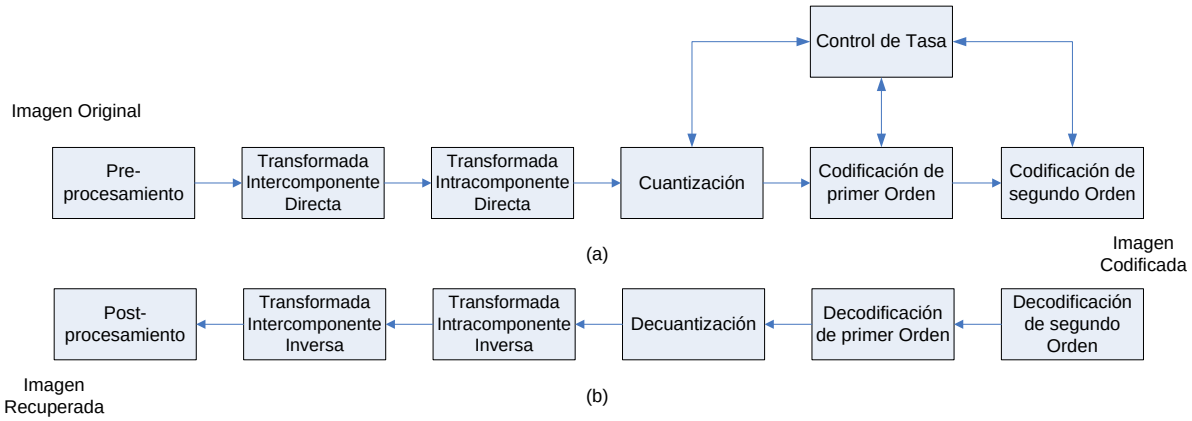

Figura 5. Estructura del códec. La estructura del (a) *Codificador y* (b) *Decodificador.*

B.4.5. Pre-procesamiento/Post-procesamiento

El códec espera que sus datos de entrada tengan un rango dinámico que está centrado aproximadamente en cero. La etapa de pre-procesamiento simplemente se asegura que este requerimiento se cumpla. Supongamos que un componente en general tiene P bits por muestra. Las muestras pueden tener o no signo, por lo que pueden tener un rango dinámico nominal de $[-2^{p-1}, 2^{p-1}-1]$ o $[0, 2^{p}-1]$, respectivamente. Si los valores de muestreo son positivos, el rango dinámico nominal está claramente centrado en un punto alejado de cero. Por lo tanto el rango dinámico nominal es ajustado restándole 2^{p-1} a cada muestra. Si los valores de un componente tienen signo, el rango dinámico nominal ya está centrado en cero, y no requiere de ningún procesamiento. Al asegurar que el rango dinámico nominal esté centrado en cero, se pueden hacer muchas suposiciones que simplifican el diseño del códec.

La parte de post-procesamiento esencialmente invierte el proceso de pre-procesamiento del codificador. Si los valores de cada componente no tenían signo, se hace una restauración del rango nominal previo. Por último, en el caso de la codificación con pérdidas, se hace un recorte para que las muestras no excedan los límites del rango permitido.

B.4.6. Transformada Intercomponente

En el codificador, la etapa de pre-procesamiento es seguida por la etapa de transformada intercomponente. En este lugar una transformada intercomponente se aplica a los datos de cada componente de tesela. Tal transformada opera en todos los componentes, y funciona para reducir la correlación entre componentes, mejorando la eficiencia al comprimir.

Dos transformadas intercomponente han sido definidas en el códec JPEG2000: la transformada de colores irreversible (*Irreversible Color Transform – ICT*) y la transformada de colores reversible (*Reversible Color Transform – RCT*). La ICT es irreversible de naturaleza real a real, mientras que la RCT es reversible y de naturaleza entero a entero. Ambas transformadas cambian la información de los datos de RGB a YCrCb. Las transformadas están diseñadas para operar en los tres primeros componentes de la imagen, asumiendo que los componentes 0, 1 y 2 corresponden a los planos rojo, verde y azul. Debido a la naturaleza de estas transformadas los componentes donde operan debes estar muestreados a la misma resolución (entiéndase, deben tener el mismo tamaño). Como consecuencia de las restricciones previas, las ICT y RCT sólo pueden ser utilizadas cuando la imagen codificada tiene al menos tres componentes, y los tres componentes estén muestreados a la misma resolución. La ICT se puede usar sólo en la compresión con pérdidas, mientras que la RCT puede ser usada en las compresiones con o sin pérdidas. Inclusive cuando una transformada puede ser empleada, no es necesario hacerlo. Entiéndase, la decisión de usar una transformada entre componentes se deja a la discreción del codificador. Después de la etapa de transformada intercomponente, los datos en cada componente se tratan independientemente.

La ICT no es más que la transformada de RGB a YCrCb clásica. La transformada se define como:

$$\begin{bmatrix} V_0(x,y) \\ V_1(x,y) \\ V_2(x,y) \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} U_0(x,y) \\ U_1(x,y) \\ U_2(x,y) \end{bmatrix} \quad (2)$$

Donde $U_0(x,y), U_1(x,y)$, y $U_2(x,y)$ son los componentes de entrada correspondientes a los planos rojo, verde y azul, respectivamente, y $V_0(x,y), V_1(x,y)$, y $V_2(x,y)$ son los componentes de salida correspondientes a los planos Y, Cr y Cb, respectivamente. La transformada inversa para esta operación es:

$$\begin{bmatrix} U_0(x,y) \\ U_1(x,y) \\ U_2(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34413 & -0.71414 \\ 1 & -1.772 & 0 \end{bmatrix} \begin{bmatrix} V_0(x,y) \\ V_1(x,y) \\ V_2(x,y) \end{bmatrix} \quad (3)$$

La RCT es una aproximación de la ICT en el plano de entero a entero (similar a la propuesta en [28]), donde la transformada directa se define como:

$$V_0(x,y) = \left\lfloor \tfrac{1}{4} \left(U_0(x,y) + 2U_1(x,y) + U_2(x,y) \right) \right\rfloor \quad (4a)$$
$$V_1(x,y) = U_2(x,y) - U_1(x,y) \quad (4b)$$
$$V_2(x,y) = U_0(x,y) - U_1(x,y) \quad (4c)$$

Con la mismas definiciones de los términos. La transformada inversa de la RCT es:

$$U_1(x,y) = V_0(x,y) - \left\lfloor \tfrac{1}{4} \left(V_1(x,y) + V_2(x,y) \right) \right\rfloor \quad (5a)$$
$$U_0(x,y) = V_2(x,y) + U_1(x,y) \quad (5b)$$
$$U_2(x,y) = V_1(x,y) + U_1(x,y) \quad (5c)$$

La etapa de transformada intercomponente inversa esencialmente invierte los efectos de la etapa de transformada intercomponente directa en el codificador. Si una transformada multicomponente fue aplicada durante la codificación, la inversa se aplica aquí. Sin embargo, a menos que la transformada sea reversible, se puede apenas llegar a una aproximación debido a la aritmética de precisión finita.

B.4.7. Transformada Intracomponente

Siguiendo la etapa de transformada intercomponente va la etapa de transformada intracomponente. En esta etapa, las transformadas operantes en los componentes separados son aplicadas. El operador utilizado en esta etapa es la transformada wavelet. En la aplicación de la transformada wavelet, ese componente es dividido en diversas subbandas.

Debido a las propiedades estadísticas de las señales de subbanda, la codificación de los datos transformados se hace más eficiente.

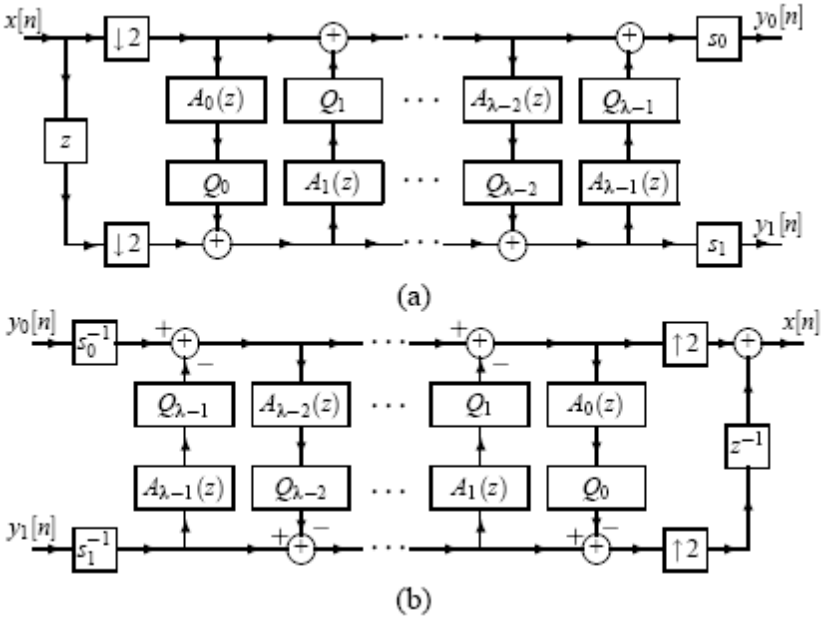

Figura 6. Diseño de un UMDFB de 2 canales I-D. (a) *Segmento de análisis.* (b) *Segmento de síntesis.*

Tanto la transformada reversible de entero a entero como la irreversible de real a real [31, 32, 36–38] son manejadas por el códec básico. El bloque básico para estas transformadas es el banco de filtros con decimado máximo uniforme de reconstrucción perfecta (*Perfect-Reconstruction Uniformly-Maximally-Decimated Filter Bank – PR-UMDFB)* de dos canales en una dimensión, cuya forma general es mostrada en la *figura 6*. Aquí nos enfocamos en el funcionamiento del UMDFB [39, 40], el cual puede ser usado para las transformadas wavelet reversibles e irreversibles usadas en el códec básico. De hecho, por el funcionamiento característico de este banco de filtros, es muy probable que esta estrategia pueda ser empleada en muchos códec. La etapa de análisis del UMDFB, mostrado en la *figura 6*(a), está asociada con la transformada directa, mientras que la etapa de síntesis, mostrado en la *figura 6*(b), está asociada con la transformada inversa. En el

diagrama, los $\{A_i(z)\}_{i=0}^{\lambda-1}, \{Q_i(x)\}_{i=0}^{\lambda-1}$, y $\{s_i\}_{i=0}^{1}$ son los parámetros de transferencia del filtro, operadores de cuantización, y ganancia (escalar), respectivamente. Para obtener mapeos de entero a entero, los $\{Q_i(x)\}_{i=0}^{\lambda-1}$ son seleccionados de tal manera que solo entreguen valores enteros, y los $\{s_i\}_{i=0}^{1}$ se eligen como enteros. Para mapeos de real a real, los $\{Q_i(x)\}_{i=0}^{\lambda-1}$ son elegidos como una identidad, y los $\{s_i\}_{i=0}^{1}$ son números reales. Para facilitar el filtrado de en los extremos de la señal, se usan métodos de extensión simétrica. Como la imagen es una señal de dos dimensiones, claramente se necesita un 2-D UMDFB. Al aplicar los UMDFB de una dimensión en las direcciones horizontal y vertical, se consigue efectivamente una 2-D UMDFB. La transformada wavelet entonces es calculada al pasar recursivamente el 2-D UMDFB a la señal pasa-bajo de las subbandas obtenida en cada nivel de descomposición.

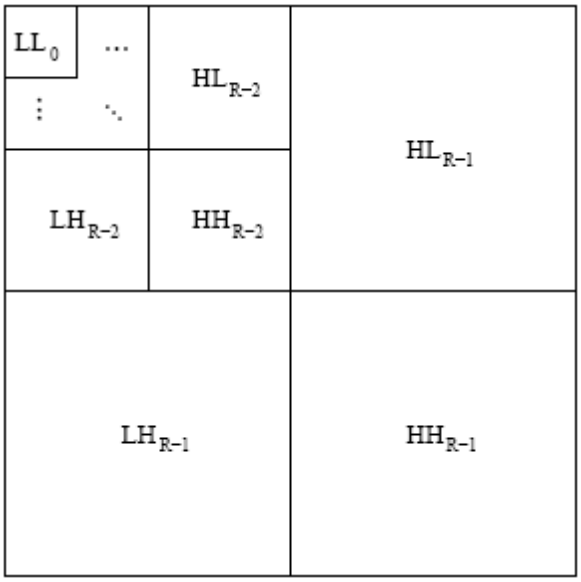

Figura 7. Estructura de subbandas.

Supongamos que se emplea una transformada wavelet de $(R-1)$ niveles. Para calcular la transformada directa, se aplica la etapa de análisis del 2-D UMDFB a los datos de cada componente de tesela iterativamente, entregando un número de señales de subbanda como resultado. Cada aplicación del segmento de análisis del 2-D UMDFB genera 4 subbandas: 1) pasabajo horizontal y vertical (LL), 2) pasabajo horizontal y pasaalto vertical (LH), 3)

pasaalto horizontal y pasabajo vertical (HL), y 4) pasaalto horizontal y vertical (HH). Una transformada wavelet de $(R-1)$ niveles se asocia con R niveles de resolución, numerados de 0 a $(R-1)$, con 0 y $(R-1)$ siendo las resoluciones mas gruesas y finas, respectivamente. Cada subbanda de la descomposición es identificada por su orientación (LL, LH, HL, HH) y por su nivel de resolución correspondiente (0, 1,…, $R-1$). La señal de entrada del componente de tesela se considera como la banda LL_{R-1}. En cada nivel de resolución (hasta el último) se descompone adicionalmente a banda LL. Por ejemplo, la banda LL_{R-1} se descompone para entregar las bandas $\mathrm{LL}_{R-2}, \mathrm{LH}_{R-2}, \mathrm{HL}_{R-2}$ y HH_{R-2}. Luego, en el siguiente nivel, la banda LL_{R-2} es descompuesta, y este procedimiento se repite hasta llegar a la banda LL_0, y se organizan los resultados como se muestra en la figura 7. En el caso donde no haya transformada, $R=1$ y solo tendríamos una subbanda (la subbanda LL_0).

Como fue previamente descrito, la descomposición wavelet está asociada a R diferentes resoluciones. Supongamos que las muestras limitantes superior izquierda e inferior derecha de un componente de tesela tienen como coordenadas (tcx_0, tcy_0) y (tcx_1-1, tcy_1-1), respectivamente. Siendo este el caso, las muestras limitantes superior izquierda e inferior derecha de un componente de tesela a una resolución r tienen de coordenadas (trx_0, try_0) y (trx_1-1, try_1-1), respectivamente, dadas por:

$$(\mathrm{trx}_0, \mathrm{try}_0) = \left(\left\lceil \mathrm{tcx}_0/2^{R-r-1} \right\rceil, \left\lceil \mathrm{tcy}_0/2^{R-r-1} \right\rceil\right) \qquad (6a)$$

$$(\mathrm{trx}_1, \mathrm{try}_1) = \left(\left\lceil \mathrm{tcx}_1/2^{R-r-1} \right\rceil, \left\lceil \mathrm{tcy}_1/2^{R-r-1} \right\rceil\right) \qquad (6b)$$

Donde r es la resolución a la que queremos trabajar. Por lo tanto, la información de un componente de tesela en una resolución en particular tiene como tamaño $(trx_1 - trx_0) \times (try_1 - try_0)$.

No sólo los sistemas de coordenadas de los niveles de resolución tienen importancia, también los sistemas de coordenadas de cada subbanda. Supóngase que notamos las coordenadas de las muestras limitantes superior izquierda e inferior derecha de cada subbanda como (tbx_0, tby_0) y (tbx_1-1, tby_1-1), respectivamente, dadas por:

$$(\mathrm{tbx}_0, \mathrm{tby}_0) = \begin{cases} \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} \right\rceil\right) & \text{for LL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} \right\rceil\right) & \text{for HL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for LH band} \\ \left(\left\lceil \frac{\mathrm{tcx}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_0}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for HH band} \end{cases} \quad (7a)$$

$$(\mathrm{tbx}_1, \mathrm{tby}_1) = \begin{cases} \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} \right\rceil\right) & \text{for LL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} \right\rceil\right) & \text{for HL band} \\ \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for LH band} \\ \left(\left\lceil \frac{\mathrm{tcx}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil, \left\lceil \frac{\mathrm{tcy}_1}{2^{R-r-1}} - \frac{1}{2} \right\rceil\right) & \text{for HH band} \end{cases} \quad (7b)$$

Donde r es el nivel de resolución al que pertenece la banda, R es el número de niveles de resolución, y $tcx_0, tcy_0, tcx_1, tcy_1$ están definidas en (1). Por lo tanto, una banda en particular tiene un tamaño de $(tbx_1 - tbx_0) \times (tby_1 - tby_0)$. De las anteriores ecuaciones podemos deducir que las coordenadas de un nivel de resolución son iguales a las coordenadas de la banda LL_r, como es de esperarse. (Mostrando que la banda LL_r es una versión de resolución reducida de los datos originales.) Como se verá posteriormente, los sistemas de coordenadas para las diferentes resoluciones y subbandas de un componente de tesela juegan un papel importante en el comportamiento del códec.

Al examinar (1), (6), y (7), observamos que las coordenadas de la muestra superior izquierda de una subbanda particular, notadas como (tbx_0, tby_0), son determinadas parcialmente por los parámetros $XOsiz$ y $YOsiz$ de la malla de referencia. A cada nivel de la descomposición, la paridad (si es par o impar) de tbx_0 y tby_0 afecta el resultado del proceso de análisis. Por eso los parámetros $XOsiz$ y $YOsiz$ tienen un efecto sutil, pero importante en el cálculo de la transformada.

Al haber descrito el entorno general de la transformada, describimos las dos transformadas wavelet soportadas por el codificador básico: las transformadas 5/3 y 9/7. La transformada

5/3 es reversible, de entero a entero, y no lineal. Esta transformada fue propuesta en [31], y es una aproximación a la transformada wavelet propuesta en [41]. La transformada 5/3 tiene un 1-D UMDFB con los parámetros:

$$\lambda = 4, \quad A_0(z) = \alpha_0(z+1), \quad A_1(z) = \alpha_1(1+z^{-1}), \qquad (9)$$
$$A_2(z) = \alpha_2(z+1), \quad A_3(z) = \alpha_3(1+z^{-1}),$$
$$Q_i(x) = x \text{ for } i = 0,1,2,3,$$
$$\alpha_0 \approx -1.586134, \quad \alpha_1 \approx -0.052980, \quad \alpha_2 \approx 0.882911,$$
$$\alpha_3 \approx 0.443506, \quad s_0 \approx 1/1.230174, \quad s_1 = -1/s_0.$$

Como la transformada 5/3 es reversible, puede ser utilizada para las codificaciones con o sin pérdidas. La transformada 9/7, al ser irreversible, solo se puede usar para la codificación con pérdidas. El número de niveles de resolución es un parámetro de cada transformada. Un valor típico de este parámetro es seis (para una imagen lo suficientemente grande). El codificador puede transformar a todos o parte de los componentes, a su discreción.

La etapa de transformada intracomponente inversa esencialmente revierte los efectos hechos en la etapa de transformación intracomponente directa. Si una transformación fue aplicada a un componente en particular durante la codificación, en esta etapa se aplica la transformada inversa correspondiente. Por los efectos de la aritmética de precisión finita el proceso de inversión no puede ser exacto a menos que se hayan empleado transformadas reversibles.

B.4.7. Cuantización/Decuantización

En el codificador, después que los datos de cada componente de tesela hayan sido transformados (por las transformadas intercomponentes y/o intracomponente) los coeficientes resultantes son cuantizados. La cuantización permite una compresión más eficiente, al representar a los coeficientes transformados con la precisión mínima requerida para obtener el nivel deseado en la calidad de imagen. La cuantización de los coeficientes transformados es una de las dos fuentes de pérdida de información en el proceso de codificación (posteriormente se mostrará la otra fuente de pérdida de información, la supresión de los datos codificados).

Los coeficientes transformados son cuantizados usando cuantización escalar con una zona de nulidad. Un cuantizador es empleado por los coeficientes de cada subbanda, y cada cuantizador tiene un solo parámetro, el tamaño de su paso. Matemáticamente, el proceso de cuantización es definido como:

$$V(x,y) = \lfloor |U(x,y)| / \Delta \rfloor \operatorname{sgn} U(x,y) \qquad (10)$$

Donde Δ es el paso del cuantizador, $U(x,y)$ son los coeficientes de subbanda de entrada, y $V(x,y)$ denota los índices de cuantización para la subbanda. Como esta ecuación está especificada en una cláusula informativa del estándar, los codificadores no tienen que usar necesariamente esta fórmula precisa. Sin embargo es muy posible que muchos codificadores, de hecho, usen la ecuación previamente notada.

El códec básico tiene dos modos distintos de operación, referidos de ahora en adelante como el modo entero y el modo real. En el modo entero, todas las transformadas empleadas son de naturaleza entero a entero (RCT, 5/3 WT). En el modo real, las transformaciones de real a real son empleadas (ICT, 9/7 WT). En el modo entero, el pazo de los tamaños de cuantización se ajusta a uno, ignorando la cuantización y forzando a los índices de cuantización a ser iguales a los coeficientes de las transformadas. En este caso, la codificación con pérdidas es posible, pero usando controles de tasa con otro mecanismo. En el modo real (que implica una codificación con pérdidas), los tamaños de cuantización son elegidos junto con el control de tasa. Hay numerosas estrategias para elegir estos tamaños, discutidos posteriormente.

Como se puede esperar, los tamaños en los pasos del cuantizados son entregados al decodificados en la información de la cabecera. También hay que notar que los tamaños en los pasos especificados en la cabecera son relativos y no absolutos. Entiéndase, el tamaño de los pasos de cuantización para cada banda está especificado relativamente al rango dinámico de la señal de subbanda.

En el decodificador, la etapa de decuantización intenta deshacer los efectos de la cuantización. Este proceso, sin embargo, no es usualmente reversible, y está sujeta a una pérdida de información. Los valores de los coeficientes de transformada se obtienen

mediante los índices de cuantización. Matemáticamente, el proceso de decuantización está definido como:

$$U(x,y) = (V(x,y) + r\,\mathrm{sgn}V(x,y))\,\Delta \qquad (11)$$

Donde Δ es el paso de cuantización, r es un parámetro fijo, $V(x,y)$ son los índices de cuantización para cada subbanda y $U(x,y)$ es la señal de subbanda reconstruida. Aunque el valor de r no está especificado normativamente en el estándar, es probable que muchos decodificadores usen el valor de un medio.

B.4.8. Codificación de Primer Orden.

Una vez que se realiza la cuantización en el codificador, entra en juego la codificación de primer orden. Esta es la primera de dos etapas de codificación. Los índices de cuantización de cada subbanda son particionadas en bloques de código. Los bloques de código son de forma rectangular, y su tamaño nominal es un parámetro libre en el proceso de codificación, sujeta a ciertas restricciones, que son: 1) el ancho y alto nominal de un código de bloque debe ser un entero potencia de dos, y 2) el producto del ancho y alto nominal no puede exceder 4096.

Suponga que el tamaño de los bloques de código es elegido tentativamente a ser $2^{xcb} \times 2^{ycb}$. En la codificación de segundo orden, a ser discutida posteriormente, los bloques de código son agrupados en precintos. Como los bloques de código no pueden cruzar los límites de un precinto, una reducción en el tamaño de los bloques puede requerirse si el tamaño de un precinto es suficientemente pequeño. Suponga que el tamaño nominal de los bloques de código después de tal ajuste es $2^{xcb`} \times 2^{ycb`}$ donde $xcb` \leq xcb$ y $ycb` \leq ycb$. La subbanda es particionada en bloques de código al superponer la subbanda con una malla rectangular con los espaciamientos de $2^{xcb`}$ y $2^{ycb`}$, respectivamente, como es mostrado en la *figura 8.* El origen de esta malla se fija en $(0,0)$ en el sistema de coordenadas de la subbanda. Un tamaño típico de códigos de bloque es 64×64 (entiéndase $xcb = 6, ycb = 6$).

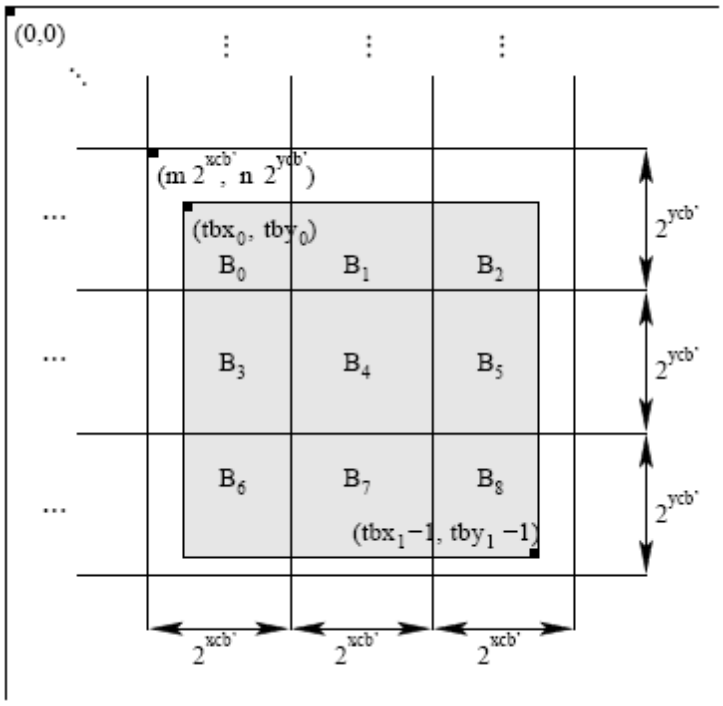

Figura 8. Partición de una subbanda en bloques de código.

Denotemos las coordenadas de la muestra superior izquierda como (tbx_0, tby_0). Como se explicó anteriormente, las coordenadas (tbx_0, tby_0) son parcialmente determinadas por los parámetros $XOsiz$ y $YOsiz$ de la malla de referencia. A su vez, los parámetros (tbx_0, tby_0) afectan la posición de los límites en los bloques de código dentro de una subbanda. De esta manera los parámetros $XOsiz$ y $YOsiz$ tienen un efecto importante en el proceso de codificación de primer orden (entiéndase, afectan la posición de los límites entre los bloques de código).

Después que la subbanda es particionada en bloques de código, cada uno de los bloques de código es codificado independientemente. La codificación se hace usando el codificador de planos de bits descrito en la sección J. Por cada bloque de código, un código embebido se produce, compuesto de varias etapas de codificación. La salida de la codificación de primer orden es una colección de datos de codificación de cada bloque de código.

En la etapa de decodificación, los pasos de codificación de los bloques son la entrada al decodificador de primer orden, estos datos son decodificados y los planos resultantes son ensamblados en subbandas. De esta manera, obtenemos los índices de cuantización de cada

subbanda. En el caso de la codificación con pérdidas, los índices de cuantización reconstruidos pueden ser aproximaciones a los índices de cuantización originales. Esto es atribuible al hecho de que el código puede incluir un subconjunto de los pasos de codificación generados en el proceso de codificación de primer orden. En la codificación sin pérdidas, los índices de cuantización deben ser los mismos que los índices originales en el lado de la codificación, como todos los pasos de codificación deben ser incluidos para la codificación sin pérdidas.

B.4.9. Codificación de Planos de Bits

El proceso de codificación de primer orden es esencialmente la codificación de planos de bits, después de que todas las subbandas han sido particionadas en bloques de código, cada bloque de código resultante es codificado independientemente usando un codificador de planos de bits. Aunque la técnica de codificación de planos de bits empleada es similar a las técnicas usadas en los árboles de ceros anidados (*Embedded Zerotree Wavelet – EZW*) [42] y en la partición en árboles jerárquicos (*Set Partitioning in Hierarchical Trees – SPIHT*), hay dos diferencias notables: 1) no hay dependencias entre bandas, y 2) hay tres pasos de codificación por plano de bits en vez de dos. La primera diferencia se basa en el hecho de que cada bloque de código está totalmente contenido en una única subbanda, y que cada bloque de código está codificado independientemente uno del otro. Al no explotar las dependencias ente subbandas, se puede conseguir una mejora en la inmunidad a errores. La segunda diferencia es menos fundamental. Al usar tres pasos de codificación en vez de dos se reduce la cantidad de datos asociados con cada paso de codificación, facilitando un control más firme sobre la tasa. Además, el usar un paso adicional por plano de bits permite dar prioridad a los datos importante, mejorando la eficiencia de codificación.

Como se notó anteriormente, hay tres pasos de codificación por plano de bits. Por orden, estos pasos son los siguientes: 1) significancia, 2) refinamiento y 3) limpieza. Todos los tres pasos de codificación leen las muestras de un bloque de código en el mismo orden como es mostrado es la *figura 10*. El bloque de código es dividido en tiras horizontales, cada una con un alto nominal de cuatro muestras. Si la altura del bloque de código no es un múltiplo de 4, la altura de la tira inferior tendrá un tamaño menor al nominal. Como se

muestra en el diagrama, las tiras se leen de arriba abajo. Dentro de una tira, las columnas se leen de izquierda a derecha. Dentro de una columna, las muestras se leen de arriba abajo.

El proceso de codificación genera una secuencia de símbolos por cada paso de codificación. Algunos o todos estos símbolos pueden ser codificados de su entropía. Para los propósitos de la codificación de entropía, se ha elegido un codificador aritmético binario adaptativo basado en el contexto – más específicamente el codificador MQ del estándar JBIG2 [35]. Por cada paso, todos los símbolos don codificados simple o aritméticamente. Los procesos de codificación simple y aritmética aseguran que unos patrones de bits nunca ocurran en la salida, permitiendo el que estos patrones sean usados con propósitos de corrección de errores.

Los pasos de limpieza siempre emplean codificación aritmética. En el caso de los pasos de significancia y refinamiento, existen dos posibilidades, dependiendo si está activado el modo de codificación de canal aritmético (también conocido como modo perezoso). Si está activado el modo perezoso, sólo los pasos de significancia y refinamiento de los cuatro planos de bit más significativos usan codificación aritmética, mientras que el resto usan codificación simple. De otra manera, todos los pasos de significancia y refinamiento usan codificación aritmética. El modo perezoso reduce significativamente la complejidad computacional de la codificación de los planos de bits, al reducir el número de símbolos que son codificados aritméticamente. Por supuesto, esto se consigue sacrificando eficiencia de codificación.Como fue indicado anteriormente, los datos de los pasos de código pueden ser codificados usando uno de los dos esquemas (codificación simple o aritmética). Los pasos de codificación consecutivos que comparten el mismo esquema de codificación constituyen un segmento. Todos los pasos de codificación en un segmento pueden formar una única palabra de código o cada paso de codificación puede formar una palabra de código. El caso elegido es determinado por el modo de terminación en efecto. Hay dos modos de terminación definidos: terminación por paso y terminación por segmento. En el primer caso todos los pasos de codificación generan una palabra de código, en el segundo, cada segmento forma una palabra de código. El terminar todos los pasos de codificación facilita una detección de errores a cambio de eficiencia de codificación.

Como se emplea una codificación aritmética basada en contexto, es necesario idear un método de selección de contexto. Hablando en general, la selección de contexto se consigue al examinar la información de los 4 u 8 vecinos de una muestra, como se ve en la *figura 9*.

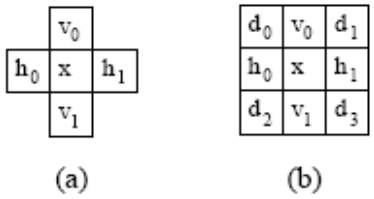

Figura 9. Esquema para la selección de contexto (a) *4 vecinos y* (b) *8 vecinos.*

En la explicación de los pasos de codificación, nos enfocaremos en el codificador ya que facilita el entendimiento del algoritmo. Los algoritmos de decodificación invierten exactamente el proceso empleado en el lado del codificador.

J.1. Paso de Significancia.

El primer paso de codificación para cada plano de bit es el paso de significancia. Este paso sirve para predecir la significancia y (como sea requerido) marcar las muestras que no hayan sido encontradas significativas aun y se espera que sean significativas en ese plano de bits. Las muestras en el bloque de código son muestreadas en el orden mostrado en la figura 10. Si una muestra no ha sido encontrada como significativa aun, y se predice el ser significativa, la significancia de la muestra es codificada con un único símbolo binario. So la muestra también resulta ser significativa, su magnitud es codificada con un único símbolo binario. En forma de pseudocódigo, el paso de significancia está descrito en el algoritmo 1.

Algoritmo 1: algoritmo de paso de significancia

```
1:  por cada muestra en el bloque de código haga
2:  si la muestra es previamente insignificante y
    Está predicha como significativa en este plano de bits entonces
3:  codificar la significancia de la muestra /*1 símbolo binario*/
4:  si la muestra es significativa entonces
5:  codificar el signo de la muestra /*1 símbolo binario*/
6:  fin si
7:  fin si
8:  fin por
```

Si el plano de bits más significativo está siendo procesado, todas las muestras se predicen insignificantes. De otra manera, una muestra se predice ser significativa si alguna muestra en su vecindad ha mostrado ser significativa. Como consecuencia de esta política de predicción, los pasos de significancia y refinamiento siempre están vacios y no son codificados explícitamente.

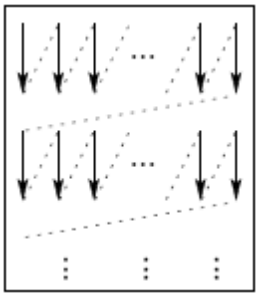

Figura 10. Orden de escaneo en un bloque de código.

Los símbolos generados durante el paso de significancia pueden o no ser codificados aritméticamente. Si se emplea una codificación aritmética, el símbolo binario que representa la información codificada es codificada usando uno de nueve contextos. El contexto usado es seleccionado basándose en la significancia de los 8 vecinos de la muestra y la orientación de la subbanda en donde la muestra está asociada (LL, LH, HL, HH). En el caso de que se use codificación aritmética, la magnitud de una muestra es codificada como la diferencia entre la magnitud real y la predicha. De otra manera la magnitud se codifica directamente. La predicción de la magnitud está hecha usando la significancia e información de magnitud de los 4 vecinos a la muestra.

J.2. Paso de Refinamiento

El segundo paso de codificación de cada plano de bits es el paso de refinamiento. Este paso señaliza a los bits después de los bits más significativos en cada muestra. Las muestras de los bloques de códigos son leídas como se mostró en la *figura 10*. Si una muestra fue encontrada como significativa en un plano de bits, el siguiente bit más significativo se

representa con un único símbolo binario. Este proceso de describe en forma de pseudocódigo en el algoritmo 2:

Algoritmo 2: Algoritmo de paso de refinamiento

```
1:  por cada muestra en el bloque de código haga
2:  si la muestra se consideraba significativa en el plano de bits previo entonces
3:  codificar el siguiente bit más significativo en la muestra /*1 símbolo binario*/
4:  fin si
5:  fin por
```

Como el paso de significancia, los símbolos pueden o no ser codificados aritméticamente. Si se emplea una codificación aritmética, cada símbolo de refinamiento es codificado en uno de tres contextos. El contexto a emplear es seleccionado en la significancia de los 8 vecinos de la muestra y la magnitud del segundo bit más significativo.

J.3. Paso de Limpieza

El tercer paso de cada plano de bits es el paso de limpieza. Este paso es usado para entregar la significancia y (si es necesario) magnitud de aquellas muestras que no han sido notadas como significantes y se predice que serán insignificantes durante el procesamiento del plano de bits.

Conceptualmente, el paso de limpieza no es muy distinto del paso de significancia. La principal diferencia radica en que el paso de limpieza entrega la información de las muestras que se predicen que serán insignificantes, en vez de aquellas que se predicen el ser significante. Algorítmicamente, sin embargo, hay una diferencia muy importante dentro de los pasos de significancia y limpieza. En el caso del paso de limpieza, las muestras son procesadas como grupos, en vez de ser procesadas individuamente como con el paso de significancia.

Recordemos el patrón de lectura de un bloque de código mostrado en la *figura 10*. Un bloque de código es dividido en tiras con una altura nominal de 4 muestras. Después, las tiras son leídas de arriba abajo, y las columnas de una tira son leídas de izquierda a derecha. Por conveniencia, nos referiremos a cada columna dentro de una tira como un escaneo

vertical. Entiéndase, cada flecha vertical corresponde a un escaneo vertical. Como será explicado posteriormente, el paso de limpieza es más fácil de entender desde la perspectiva de un escaneo vertical (y no solo por cada muestra en individual).

El paso de limpieza procesa cada uno de los escaneos verticales en el orden establecido, con cada escaneo vertical siendo procesado como sigue. Si el escaneo vertical tiene cuatro muestras (entiéndase, es un escaneo completo), se necesita información de significancia para estas muestras, y todas las muestras se predicen insignificantes, un modo especial, llamado modo de agregación, se establece. En este modo, el número de muestras insignificantes en el escaneo vertical son codificadas. Luego, las muestras cuya información de significancia es representada por agregación son saltadas, y se procesan las muestras restantes exactamente como en el paso de significancia. En pseudocódigo, este proceso es descrito por el algoritmo 3:

Algoritmo 3: algoritmo de paso de limpieza

```
1:   por cada lectura vertical en el bloque de código haga
2:   si las cuatro muestras en el escaneo son previamente insignificantes y no han sido visitadas
     previamente y no tienen vecinos significativos entonces
3:   codificar el número de muestras insignificantes iniciales por agregación
4:   ignorar toda muestra indicada como insignificante por agregación
5:   fin si
6:   mientras haya más muestras que procesar en un escaneo vertical haga
7:   si la muestra era previamente insignificante y no ha sido considerada entonces
8:   codificar el significancia de la muestra si no ha implicada en el proceso /*1 símbolo binario*/
9:   si la muestra es significativa entonces
10:  codificar el signo de la muestra /*1 símbolo binario*/
11:  fin si
12:  fin si
13:  fin mientras
14:  fin por
```

Cuando se activa el modo de agregación, las cuatro muestras del escaneo vertical son examinadas. Si las cuatro muestras son insignificantes, un símbolo indicando esto se codifica, y el proceso del escaneo vertical es completado. De otra manera, un símbolo indicando significancia se codifica, y dos símbolos binarios se usan para codificar las primeras muestras insignificantes en el escaneo vertical.

Los símbolos generados durante el paso de limpieza siempre son codificados aritméticamente. En el modo de agregación, el símbolo de agregación es codificado usando un único contexto, y los dos símbolos de longitud de insignificancia son codificadas usando

un único contexto con una distribución de probabilidad uniforme. Cuando el modo de agregación no es empleado, la codificación de significancia y magnitud funciona como el paso de significancia.

B.4.10. Codificación de Segundo Orden

En el codificador, la codificación de primer orden es seguida por la codificación de segundo orden. La entrada en el proceso de codificación de segundo orden son los grupos de pasos de codificación de planos de bits generados en la codificación de primer orden. En la codificación de segundo orden, la información de los pasos de codificación es empaquetada en unidades de datos llamados paquetes, en un proceso conocido como paquetización. Los paquetes resultantes son luego la salida para el código final. El proceso de paquetización impone una organización particular de los datos de los pasos de codificación dentro del código final. Esta organización facilita muchas de las habilidades deseadas del códec incluyendo la escalabilidad de tasa y la recuperación progresiva por fidelidad o resolución.

Un paquete no es nada más que una colección de datos de los pasos de codificación. Cada paquete comprende de dos partes: un encabezado y un contenido. El encabezado indica que pasos de codificación están incluidos en el paquete, mientras que el contenido posee los datos de paso de codificación de por sí. En el código final, el encabezado y el contenido pueden aparecer juntos o por separado, dependiendo de las opciones de codificación a ser efectuadas.

La escalabilidad de tasas se obtiene a través de capas de calidad. Los datos codificados de cada tesela son organizadas en L capas, numeradas desde 0 hasta $L-1$, donde $L \geq 1$. Cada paso de codificación es entonces asignado a una de las L capas o se descarta. Los pasos de codificación conteniendo los datos más importantes son incluidos en las capas bajas, mientras que los pasos de codificación relacionadas con los detalles más finos son incluidos en las capas superiores. En la decodificación, la calidad de la imagen mejora incrementalmente con cada capa procesada. En el caso de la compresión con pérdidas, algunos pasos de codificación pueden ser descartados (entiéndase, no incluidos en ninguna capa) en cuyo caso el control de tasa debe decidir qué pasos incluir en el código final. En la

codificación sin pérdidas, todos los pasos de codificación deben ser incluidos. Si se emplean múltiples capas (entiéndase, $L > 1$), el control de tasa decide qué pasos de codificación se incluyen en cuales capas. Como ciertos pasos de codificación pueden ser descartados, la codificación de segundo orden es la segunda fuente controlada de pérdida de información en el proceso de codificación.

Recordemos de la sección I, que cada paso de codificación está asociado con un componente, nivel de resolución, subbanda, y bloque de código en particular. En la codificación de segundo orden, un paquete es generado por cada componente, nivel de resolución, capa, y precinto. Un paquete no necesariamente necesita contener información. Los paquetes vacíos son necesarios a veces porque un paquete debe ser generado por cada grupo de componente-resolución-capa-precinto aunque el paquete no entregue información nueva.

Como se mencionó en la sección G, un precinto es esencialmente un grupo de bloques de código derivadas de una partición de la banda *LL* madre (entiéndase, la banda *LL* en el nivel de resolución siguiente). Cada nivel de resolución tiene un tamaño de precinto nominal. El ancho y alto nominal de cada precinto debe ser una potencia de 2, sujeta a ciertas restricciones (el tamaño máximo del ancho y alto es 2^{15}). La banda *LL* asociada con cada nivel de resolución es dividida en precintos. Esto de logra al superponer la banda *LL* con una malla regular con espaciamientos horizontales y verticales de $2^{PPx} \times 2^{PPy}$, respectivamente. Como se muestra en la *figura 11*, donde la malla está alineada con el origen del sistema de coordenadas de la banda *LL*. Los precintos en el borde de la subbanda pueden tener dimensiones menores al tamaño nominal. Cada una de las regiones resultantes son mapeadas a cada una de sus subbandas hijas (si las hay) en el siguiente nivel de resolución. Esto se consigue haciendo la transformación de coordenadas $(u, v) = (\left[\frac{x}{2}\right], \left[\frac{y}{2}\right])$ donde (x, y) y (u, v) son las coordenadas en un punto de la banda *LL* y su subbanda hija, respectivamente. Debido a la manera en cómo se hace la partición por precintos, los límites de los precintos están siempre alineados con los límites de los bloques de código. Algunos precintos pueden estar vacíos. Supóngase que el tamaño de un código de bloque es de

$2^{xcb`} \times 2^{ycb`}$. Por lo tanto habrá $2^{PPx`-xcb`} \times 2^{PPy`-ycb`}$ grupos de bloques de código en un precinto nominalmente, donde:

$$PPx' = \begin{cases} PPx & \text{for } r = 0 \\ PPx - 1 & \text{for } r > 0 \end{cases} \quad (12)$$

$$PPy' = \begin{cases} PPy & \text{for } r = 0 \\ PPy - 1 & \text{for } r > 0 \end{cases} \quad (13)$$

Y r es el nivel de resolución.

Como los datos de paso de codificación de diferentes precintos son codificados en diferentes paquetes, al usar precintos más pequeños se reduce la cantidad de datos contenidos en cada paquete. Si hay menos información contenida en cada paquete, un bit errado puede generar menos pérdidas de información (puesto que los errores en un paquete no afectan la decodificación de los otros paquetes). Por lo tanto un tamaño de precinto pequeño genera una resistencia a errores mejorada, mientras que la eficiencia de codificación es degradada debido al número mayor de paquetes.

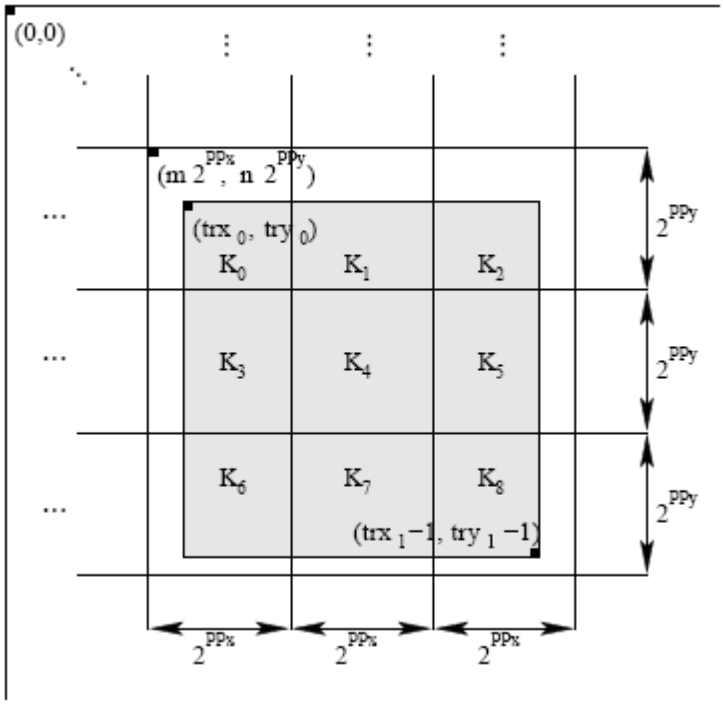

Figura 11. Partición de una resolución en recintos.

Hay más de un método implementado para ordenar internamente a los paquetes. Tales métodos son llamados progresiones. Existen cinco progresiones definidas internamente: 1) orden por capa-resolución-componente-posición, 2) orden por resolución-capa-componente-posición, 3) orden por resolución-posición-componente-capa, 4) orden por posición-componente-resolución-capa y 5) orden por componente-posición-resolución-capa. El orden de los paquetes está dado por el nombre del ordenamiento, donde la posición se refiere al número del precinto, y las claves de organización están asignadas en orden decreciente de importancia. Por ejemplo, en el caso de la primera forma de organización, los paquetes serían organizados primero por capas, luego por resolución, luego por componente, y por último por precinto. Esto corresponde a una recuperación progresiva en un escenario pensado en fidelidad. El segundo orden está pensado en la recuperación de datos progresivos por resolución. Los otros tres órdenes son in poco más esotéricos. También es posible especificar progresiones definidas por el usuario al costo de un encabezado de código mayor.

En el escenario más sencillo, todos los paquetes de una tesela en particular aparecen juntos en el mismo código. Sin embargo existen provisiones para entrelazar paquetes de diferentes teselas, permitiendo una flexibilidad mejorada en la organización de los datos. Si, por ejemplo, se deseara la recuperación progresiva de una imagen teselada, uno incluiría a todos los paquetes asociados con la primera capa de las diversas teselas, seguidos por los paquetes asociados a la segunda capa, y de allí en adelante.

En el decodificador, el proceso de decodificación de segundo orden extrae los múltiples pasos de codificación del código (entiéndase, depaquetización) y asocia cada paso de codificación a su bloque de código correspondiente. En la compresión con pérdidas, no todos los pasos de codificación pueden estar presentes, ya que el codificador pudo haber descartado algunos. En la codificación con pérdidas, todos los pasos de codificación deben estar presentes en el código.

En la siguiente sección, describimos el proceso de paquetización con mayor detalle, para facilitar nuestra comprensión, elegimos explicar este proceso desde el punto de vista del decodificador. Los algoritmos del decodificador pueden ser deducidos trivialmente conociendo los algoritmos del codificador.

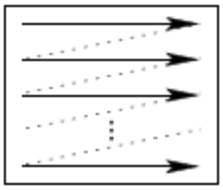

Figura 12. Escaneo de bloques de código en un precinto.

K.1. Codificación del Encabezado

El encabezado del paquete correspondiente a un componente, nivel de resolución, capa y precinto particular, es codificado de la siguiente manera. Primero, un único símbolo binario se agrega para indicar si hay algún dato de pasos de codificación en el paquete (entiéndase, si no está vacío). Si el paquete está vacío, no se requiere procesamiento posterior y el algoritmo se termina. De otra forma, se procede a examinad cada subbanda en el nivel de resolución indicado en un orden fijo.

Por cada subbanda, se visitan los bloques de código pertenecientes al precinto de interés leyéndolos en el orden mostrado en la figura 12. Para procesar un único bloque de código, empezamos determinando si hay algún dato de paso de codificación para ser incluida. Si no ha habido previos datos de paso de codificación incluidos en este bloque de código, la información de inclusión se agrega en un procedimiento de codificación de árbol cuádruple. De otra forma, un símbolo binario se emite indicando la presencia o ausencia de nuevos datos de paso de codificación en el bloque de código. Si no hay más pasos de codificación para agregar, procedemos a procesar el siguiente bloque de código en el precinto. Asumiendo que se incluyen nuevos datos de paso de codificación, continuamos el proceso del bloque de código actual. Si es la primera vez que los datos de paso de codificación son incluidos, codificamos el número de planos de bits insignificantes usando un algoritmo de codificación de árbol cuádruple. Luego, el número de pasos de codificación nuevos, y la longitud de los datos asociada con esos pasos. Unos bits de relleno se aplican a todos los datos de empaquetamiento para asegurarse que ciertos patrones de bits nunca ocurran en la

salida, permitiendo ser usados para verificación de errores. El proceso completo de codificación de encabezamientos está descrito en el algoritmo 4:

Algoritmo 4: algoritmo de codificación del encabezado de un paquete

```
si el paquete no está vacío entonces
codificar un indicador de paquete no vacío /*1 símbolo binario*/
por cada subbanda en un nivel de resolución haga
por cada bloque de código en un precinto de subbandas haga
codificar la información de inclusión /*1 símbolo binario o un árbol de etiquetas*/
si no hay más pasos de codificación entonces
saltar al siguiente bloque de código
fin si
si es la primera inclusión al bloque de código entonces
codificar el número de planos de bit insignificantes /*árbol de etiquetas*/
fin si
codificar el número de nuevos pasos de codificación
codificar un indicador de incremento de longitud
codificar la longitud de los pasos de codificación
fin por
fin por
si no
codificar indicador de paquete vacío /*1 símbolo binario*/
fin si
rellenar al límite de bytes
```

K.2. Codificación del Contenido

El algoritmo usado para codificar el contenido del paquete es relativamente simple. Los bloques de código son examinados en el mismo orden que en el caso del encabezado. Si algún paso de codificación fue especificado en el encabezado, los datos de paso de codificación son concatenados en el contenido del paquete. Este proceso es resumido en el algoritmo 5:

Algoritmo 5: algoritmo de codificación del contenido del paquete

```
por cada subbanda en un nivel de resolución haga
por cada bloque de código en un precinto de subbanda haga
si (hay nuevos pasos de codificación en el bloque de código) entonces
entregar los datos de los pasos de codificación
fin si
fin por
fin por
```

B.4.11. Control de Tasa

En el codificador, el control de tasa se obtiene con dos mecanismos distintos: 1) la elección del tamaño del paso de cuantización, y 2) la selección del subconjunto de pasos de codificación a incluirse en el código. Cuando el modo entero de codificación será usado (entiéndase, cuando solo se emplean transformadas de entero a entero) solo el segundo mecanismo puede usarse, puesto que los pasos de cuantización deben ser fijados a uno. Cuando se usa el modo de codificación real, entonces cualquiera o ambos métodos de control pueden ser usados.

Cuando se emplea el primer mecanismo, los tamaños del paso de cuantificación son ajustadas para controlar las tasas. Cuando aumenten los tamaños de paso, la tasa disminuye, a expensas de una distorsión mayor. Aunque este mecanismo de control de tasa es relativamente simple, tiene un problema potencial. Cada vez que los tamaños de paso de cuantización cambian, los índices de cuantización cambian, y se debe realizar una codificación de primer orden de nuevo. Como la codificación de primer orden requiere una cantidad considerable de procesamiento, esta aproximación al control de tasa puede no ser práctica en codificadores enfocados en eficiencia.

Cuando se usa el segundo mecanismo, el codificador puede elegir el descarte de varios pasos de codificación para controlar la tasa. El codificador sabe la contribución de cada paso de codificación a la tasa, y también puede calcular la distorsión asociada con cada paso de codificación. Usando esta información el codificador puede incluir los pasos de codificación necesarios por unidad de distorsión por tasa hasta que el número de bits han sido agotados. Esta aproximación es muy flexible puesto que muchas métricas de distorsión pueden ser acomodadas fácilmente (error cuadrático medio, error cuadrático medio visualmente pesado).

Para más información sobre el control de tasa, se puede referir a [14] y [28].

B.4.12. Codificación de Áreas de Interés

El códec está diseñado para codificar diferentes zonas de una imagen con diferente fidelidad. Esta característica es conocida como codificación de áreas de interés (*ROI*). Para

soportar la codificación ROI, una técnica simple y flexible es empleada como se describe abajo:

Cuando una imagen es sintetizada a sus coeficientes de transformación, cada coeficiente contribuye solamente a una región en la reconstrucción. Por lo tanto, una manera para codificar un ROI con mayor fidelidad que el resto de la imagen sería el identificar los coeficientes que contribuyen al ROI, y codificar algunos o todos los coeficientes con mayor precisión que los demás. Esta es la premisa básica detrás de la técnica de codificación de ROI empleada en el códec JPEG2000.

Cuando una imagen es codificada en un ROI, algunos de los coeficientes de transformada son identificados como más importantes que los otros. Los coeficientes de mayor importancia son identificados como coeficientes ROI, mientras que el resto de coeficientes son identificados como coeficientes de fondo. Hay que notar que hay una correspondencia de uno a uno entre los coeficientes de transformación y los índices de cuantización, con lo que se pueden definir los índices de cuantización para el ROI y el fondo como índices de cuantización de ROI y de fondo, respectivamente. Con esta terminología definida, estamos en la posición para describir cómo la codificación de ROI funciona con el resto de la estructura de codificación.

La funcionalidad de la codificación de ROI afecta el proceso de codificación de primer orden. En el codificador, antes de que los índices de cuantización de las múltiples subbandas sean codificadas en los planos de bits, los índices de cuantización de ROI son aumentados a una potencia de dos (haciendo una traslación de bits a la izquierda). Este aumento logra que todos los bits de los índices de cuantización de ROI estén en planos más significativos que los bits de los índices de cuantización del fondo. Como consecuencia, toda la información de los índices de cuantización de ROI sería procesada antes que la información de los índices de fondo. De esta manera, el ROI puede ser reconstruido con una fidelidad más alta que el fondo.

Antes de que los índices de cuantización sean codificados en el plano de bits, el codificador examina los índices de cuantización de fondo de todas las subbandas buscando el índice de mayor magnitud. Suponga que este índice tiene su bit más significativo en la posición $N -$

1. Todos los índices de ROI son desplazados N bits a la izquierda, y la codificación de planos de bits continua normalmente. El valor de desplazamiento de ROI N es incluido en el código.

En el proceso de decodificación, cualquier índice de cuantización con bits no nulos en el plano N o superior pertenece al grupo de ROI. Después de que los índices de cuantización son obtenidos del proceso de decodificación de planos de bits, todos los índices de ROI son rearmados a la derecha por una cantidad de N bits. Esto deshace el efecto de escalamiento en el lado del codificador.

El grupo de ROI puede ser elegido para corresponder a los coeficientes de transformada afectando una región particular en una imagen o un subconjunto de coeficientes que afectan a esa región. Esta técnica de codificación de ROI tiene un número de propiedades deseadas. Primero, el ROI puede tener cualquier forma y puede consistir de múltiples regiones aisladas. Segundo, no hay necesidad de señalizar explícitamente el grupo de ROI, puesto que puede ser deducido por el decodificador con el valor de traslación de ROI y la magnitud de los índices de cuantización.

Para más información sobre la codificación ROI, se recomienda leer [43,44].

B.4.12. Código

Para pode especificar la representación codificada de una imagen, se emplean dos niveles de sintaxis en el códec. El nivel de sintaxis más bajo está asociado con lo conocido como código base. El código base es esencialmente una secuencia de registros etiquetados y sus datos respectivos.

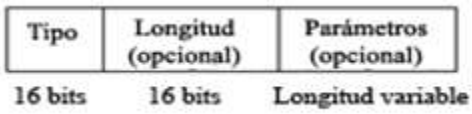

Figura 13. Estructura de los marcadores de segmento.

El armado de bloques del código base se fundamenta en los marcadores de segmento. Como se muestra en la figura 13, un marcador de segmento está compuesto de tres campos: los campos de tipo, longitud y parámetros. El campo de tipo (o marcador) identifica la clase de segmento que es en particular. El campo de longitud especifica el número de bytes en el segmento de marcadores. El campo de parámetros provee una información adicional específica al tipo de marcador. No todos los marcadores de segmento tienen campos de longitud y parámetros. La presencia (o ausencia) de estos campos está determinada por el tipo de marcador. Cada tipo de marcador tiene su propia manera de identificar su información.

Figura 14. Estructura del código.

El código básico es simplemente una secuencia de marcadores de segmento y datos auxiliares (entiéndase, la información de los paquetes) organizados como se muestra en la figura 14. El código básico consiste de un encabezado principal, seguido de los encabezados y contenidos de las teselas, seguido por un segmento de finalización. Una lista de los marcadores de segmento más importantes está dada en la tabla II. Los parámetros especificados en los marcadores de segmento en el encabezado principal funcionan como valores por defecto para el código completo. Estos valores por defecto pueden, no obstante, ser sobrescritos en una tesela particular al especificar nuevos valores en el encabezado de la tesela.

Tabla II: Tipos de Marcadores de Segmento

Tipo	**Descripción**
Start of codestream (SOC)	Muestra el comienzo del código. Siempre el primer marcador de segmento en el código (el primer marcador de segmento en el encabezado)
End of codestream (EOC)	Muestra el final del código. Siempre el último marcador de segmento.
Start of Tile-Part (SOT)	Indica el comienzo de un encabezado de tesela.
Start of Data (SOD)	Muestra el fin del encabezado de tesela. El contenido de la tesela sigue inmediatamente tras este marcador.
Image and Tile Size (SIZ)	Entrega las características básicas (tamaño, número de componentes, precisión de las muestras) y parámetros de teselado. Siempre el segundo marcador de segmento en el archivo-
Coding Style Default (COD)	Especifica los parámetros generales de codificación y decodificación.
Coding Style Components (COC)	Especifica un subconjunto de parámetros para un componente en específico.
Quantization Default (QCD)	Especifica los parámetros de cuantización.
Quantization Component (QCC)	Especifica los parámetros de cuantización de un componente específico.
Region of Interest (RGN)	Especifica los parámetros de la zona de interés (ROI).

Todos los marcadores de segmento, segmentos de paquetes y contenidos de paquete tienen una longitud que es múltiplo de 8 bits. Como consecuencia, todos los marcadores están alineados por bytes, y el código siempre será un múltiplo entero de bytes.

B.4.13. Formato del Archivo

El código básico provee la información mínima suficiente para decodificar una imagen (entiéndase, suficiente información para deducir los valores de las muestras de la imagen a decodificar). Mientras que en algunas aplicaciones sencillas esta información es suficiente, en otras aplicaciones se requiere de datos adicionales. Para mostrar una imagen decodificada, por ejemplo, es a menudo necesario el conocer características adicionales, tales como los espacios de color de los datos a decodificar, o los atributos de opacidad. Además, en algunas situaciones, es beneficioso el tener información adicional sobre una imagen (como autoría, origen, etc.) Para permitir la especificación de esta clase de datos, se agrega un nivel adicional de sintaxis al codificador. Este nivel de sintaxis es conocido como formato de archivo. El formato de archivo es usado para representar tanto a la imagen codificada como la información auxiliar de la imagen. Aunque el formato de imagen es opcional, será usada sin duda alguna por muchas aplicaciones.

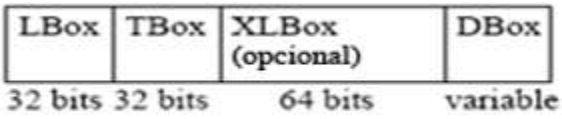

Figura 15. Estructura de una caja.

El bloque básico del formato de archivo es conocido como una caja. Como se muestra en la figura 15, una caja está compuesta nominalmente por cuatro campos: los campos LBox, TBox, XLBox, y DBox. El campo LBox especifica la longitud de la caja en bytes. El campo TBox indica el tipo de caja (entiéndase, la naturaleza de la información contenida en la caja). El campo XLBox es un indicador de tamaño extendido cuyo tamaño es muy grande para codificarse en 32 bits (el tamaño de LBox). Si el campo de LBox es 1, entonces el campo XLBox está presente y contiene la longitud total de la caja. De otra manera el

campo XLBox no existe. Como terminología, una caja que posee otras cajas en su campo de DBox es conocida como una supercaja. Los tipos de caja más importantes se incluyen en la tabla III.

Tabla III: Tipos de Marcadores de Segmento

Tipo	**Descripción**
JPEG-2000 Signature	Identifica el archivo como un formato JP2, siempre la primera caja en un archivo JP2.
File Type	Especifica la versión del formato usado por el archivo. Siempre la segunda caja en un archivo JP2.
JP2 Header	Especifica la información del archivo en general. (una supercaja)
Image Header	Especifica las dimensiones y demás características básicas de la imagen.
Color Especification	Especifica el plano de colores a los que la imagen pertenece.
Contiguous Code Stream	Contiene un código básico.

Un archivo es una secuencia de cajas. Como ciertas cajas están definidas para contener otras cajas, hay una estructura jerárquica natural en un archivo. La estructura general de un archivo es mostrado en la figura 16. La caja de identificación del JPEG2000 siempre va primero, mostrando que el archivo tiene un formato correcto. La caja de tipo de archivo siempre sigue a la primera, indicando la versión del formato con la que se ha formado este archivo. Aunque existen algunas restricciones respecto al orden de las siguientes cajas. Se permite cierta flexibilidad en este caso.la caja de encabezado contiene un grupo adicional de cajas. La caja de encabezado de la imagen especifica varias características básicas de la imagen (tamaño de la imagen, número de componentes, etc.) la caja de bits por componente indica la precisión y el signo de las muestras de cada componente. La caja de especificación de color identifica los planos de colores (para visualización) e indica cuales componentes entregan que tipo de información espectral (entiéndase, la correspondencia entre componentes y planos de color/opacidad). Todo archivo debe contener al menos una caja de código (están permitidas las cajas continuas de código para facilitar la especificación de secuencias de imágenes para animaciones triviales.) Cada caja de código contiene un código básico de contenido. De esta manera, los datos codificados de una imagen son

anidados en un archivo. Adicionalmente a las cajas mostradas, hay cajas para especificar la captura y la resolución de visualización de una imagen, información de las paletas de colores, información de propiedad intelectual, e información específica de la aplicación.

Aunque parte de esta información almacenada en el formato de archivo es redundante (porque ya ha sido especificada en el nivel de código básico), esta redundancia permite una manipulación trivial de los archivos sin conocimiento de la sintaxis del código. La extensión de archivo *"jp2"* se usará para identificar los archivos que incluyen datos en el formato de archivos JP2. Para más información sobre este formato, se puede referir a [45].

B.4.14. Extensiones

Aunque el códec básico de por si es muy versátil, hay algunas aplicaciones que pueden beneficiarse con ciertas características no presentadas en el sistema básico. Para estos fines, la parte 2 del estándar [18] define numerosas extensiones al sistema básico. Algunas de las extensiones propuestas incluyen: 1) transformadas intercomponentes adicionales (transformadas wavelet multidimensionales); 2) transformadas intracomponentes adicionales (transformación de subbanda basados en filtros arbitrarios y árboles de descomposición, filtros diferentes en los sentidos horizontales y verticales); 3) transformadas wavelet superpuestas; 4) métodos de cuantización adicionales como el método de codificación de Trellis [46,47]; 5) soporte de ROI mejorado (como un mecanismo para definir la forma del ROI y un cambio de bits arbitrario); y 6) extensiones al formato de archivo, incluyendo soporte para espacios de color adicionales y documentos compuestos.