ALGORITMO DE COMPRESIÓN Y RECONSTRUCCIÓN DE IMÁGENES FIJAS APLICANDO LA TEORÍA DE WAVELETS

YACIRO CABEZAS BURBANO
JAIRO ALEJANDRO GUEVARA COLLAZOS

Trabajo de grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Director
HAROLD A. ROMO
INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO I+D EN NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
POPAYÁN
2005

ALGORITMO DE COMPRESIÓN Y RECONSTRUCCIÓN DE IMÁGENES FIJAS APLICANDO LA TEORÍA DE WAVELETS

YACIRO CABEZAS BURBANO
JAIRO ALEJANDRO GUEVARA COLLAZOS

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO I+D EN NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
POPAYÁN
2005

NOTA DE ACEPTACIÓN

Reunido el jurado calificador, constituido por:

_____

_____

para evaluar el Trabajo de Grado titulado:

ALGORITMO DE COMPRESIÓN Y RECONSTRUCCIÓN DE IMÁGENES FIJAS APLICANDO LA TEORÍA DE WAVELETS

Bajo la dirección de:

Ing. Harold A. Romo

Y realizado por:

Yaciro Cabezas Burbano
Jairo Alejandro Guevara C.

Acordó otorgar el concepto de: APROBADO ( ) NO APROBADO ( )

JURADO COORDINADOR: _____

(Firma)

JURADO: _____

(Firma)

A mis padres
Yaciro Cabezas.

A mi familia
Jairo Guevara.

AGRADECIMIENTOS

Los autores de este proyecto expresan agradecimientos por su colaboración y ayuda a todas y cada una de las personas que de una u otra manera contribuyeron en la realización del mismo. Gracias también a la Universidad del Cauca y al Departamento de Telecomunicaciones por la formación recibida y que hoy nos convierte en unos profesionales íntegros.

Abstract

La teoría de wavelets es una muy buena herramienta para analizar señales no estacionarias que la convierte en un método con algunas ventajas sobre el tradicional análisis de Fourier. Adicional a esto, el campo de aplicación de las wavelets ha crecido en forma exponencial en los últimos años, siendo la compresión de imágenes una de las áreas de especial interés considerando su contribución al mejoramiento en el desempeño de las redes de telecomunicaciones. En este trabajo se da una introducción a la teoría de wavelets aplicada a la compresión de imágenes fijas. Para abordar esta temática el trabajo presenta inicialmente una base matemática que es el soporte para entender el funcionamiento de las wavelets. Posteriormente se presenta algunos conceptos básicos sobre imágenes digitales y se explica un sistema general de compresión de imágenes, para terminar con un desarrollo detallado sobre la aplicación implementada en Matlab, la cual comprime imágenes fijas con diferentes familias wavelets a distintos niveles, dando los respectivos resultados del rendimiento de la misma.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

1 INTRODUCCIÓN

El procesamiento digital de señales es una disciplina que desarrolla las bases teóricas y algorítmicas mediante las cuales se puede extraer información del mundo real de manera automática a partir de una señal observada, de un conjunto de señales o de una secuencia. Tal procesamiento cobra cada vez mayor importancia en muchos campos del conocimiento al tiempo que aprovecha el continuo incremento de las capacidades de los computadores, y es por ello que se ha venido estudiando mediante diferentes técnicas desde hace muchos años, dentro de las cuales, la que mayor incidencia ha tenido en su estudio es la herramienta matemática.

El matemático francés, llamado Joseph Fourier fue quien proporcionó el mayor aporte al procesamiento digital de señales estableciendo que una señal o función podía ser representada como la suma, posiblemente infinita, de senos y cosenos (o en forma equivalente como exponenciales complejas). Hoy en día la teoría de Fourier, es conocida como la transformada de Fourier [1], la cual es ampliamente utilizada en la solución de problemas científicos y de ingeniería en diferentes campos, tales como física cuántica, óptica, electrónica, astronomía, acústica y muchos otros.

Por medio de la transformada de Fourier, las señales pueden ser interpretadas como una combinación lineal de ondas armónicas o tonos puros, por lo que se observa de una manera casi intuitiva que la señal en un instante de tiempo es reemplazada por la suma de varios tonos puros. Para poder expandir o representar una señal en una forma equivalente o aproximada, la transformada de Fourier utiliza dos funciones base que son seno y coseno, las cuales tienen ciertas características como su suavidad, no presentan discontinuidades y

no son localizables en el tiempo[1], la representación individual de una frecuencia, entre otras, lo que hace que esta transformada sea extremadamente útil en el análisis de fenómenos periódicos, de tiempo invariante o estacionarios. Sin embargo, ciertas señales cuya amplitud varía en forma abrupta en el tiempo o señales cuyo contenido de frecuencia es variable de un instante de tiempo a otro, más conocidas como señales no estacionarias, no son tratadas de manera adecuada mediante la transformada de Fourier debido a ciertas limitaciones de este análisis en el campo tiempo - frecuencia. Es en estos términos entonces donde entra en juego la nueva herramienta matemática llamada *wavelet* o *transformada wavelet* [1].

La aparición de la *transformada wavelet* como herramienta matemática es relativamente reciente, aunque las ideas esenciales en las que se basa ya eran objeto de análisis mucho tiempo antes de que se plasmaran de forma analítica. Se trata de una transformación lineal al igual que la transformada de Fourier, sin embargo a diferencia de ésta, proporciona la localización en el dominio del tiempo de las diferentes componentes en frecuencia presentes en una señal dada. La transformada de Fourier ventaneada consigue parcialmente la identificación tiempo - frecuencia, cuyo principio consiste en multiplicar la señal a analizar por una función de soporte compacto[2] [2] llamada ventana y luego calcular su transformada de Fourier; de modo que trasladando dicha función a lo largo de la señal, se obtiene información local de la misma en el ancho de esta ventana. Sin embargo, el tamaño fijo de la ventana supone una limitación ya que puede resultar demasiado grande para analizar frecuencias altas y demasiado pequeño para analizar frecuencias bajas.

En el caso de la *transformada wavelet*, su funcionamiento se basa en el análisis multirresolución donde se trata la señal con funciones básicas llamadas *wavelets* a diferentes resoluciones, de tal forma que las *wavelets* empleadas para analizar señales o secciones de señal de alta frecuencia son estrechas, mientras que las destinadas para analizar las de baja frecuencia son más anchas [2]. La *teoría de wavelets* se puede presentar principalmente de dos formas: la consideración de la forma integral de la

[1] Funciones con dominio $(-\infty,+\infty)$

[2] *Soporte compacto*: La función es no nula en un rango [a,b] y es nula fuera de éste.

transformada (forma continua) y la consideración del análisis multirresolución basado en filtros digitales (forma discreta) que se adapta a los procesos numéricos realizados por computador.

El término *wavelet* se define como una "pequeña onda" o función localizable en el tiempo, que desde una perspectiva de análisis o procesamiento de señal puede ser considerada como una herramienta matemática para la representación y segmentación de señales, análisis tiempo - frecuencia, y fácil implementación algoritmos computacionales. Las características propias de la *transformada wavelet* brindan la posibilidad de representar señales en diferentes niveles de resolución, representar en forma eficiente señales con variaciones abruptas (discontinuidades), analizar señales no estacionarias permitiendo saber el contenido en frecuencia de una señal y cuándo estas componentes de frecuencia se encuentran presentes en dicha señal.

Diversas revistas de carácter investigativo alrededor del mundo dan constancia de sorprendentes resultados producto del desarrollo y aplicación de la *teoría de wavelets* en los últimos años. Es así como actualmente se puede encontrar una gran variedad de estudios referentes a la aplicación de esta teoría para afrontar problemas relacionados con el tratamiento de señales. Tal es el caso del proceso de reducción de ruido en señales de audio, detección de irregularidades locales en ciertos tipos de señales (electrocardiogramas, vibraciones de motores, ...), reconocimiento de patrones, y específicamente en el ámbito de las imágenes digitales, se puede destacar su aplicación en realce de bordes, reducción de ruido, filtrado de imágenes, detección de irregularidades locales, y la compresión de imágenes fijas para su almacenamiento en formato digital, siendo ésta última quizá la de mayor renombre.

La compresión de imágenes surge debido a la necesidad de generar y almacenar grandes cantidades de información que generalmente sobrepasan la capacidad de almacenamiento y transporte que presentan los diferentes sistemas de comunicaciones empleados en la actualidad, los cuales a pesar de ser objeto de un gran desarrollo tecnológico que día a día mejora dichas capacidades, se ven continuamente saturados debido al constante incremento

de la información. A pesar de que la computación en diferentes dominios ha permitido sofisticar la naturaleza del procesamiento de las imágenes con el propósito de extraer al máximo cualquier información disponible en las mismas, la digitalización de imágenes de alta fidelidad acentúa ostensiblemente la demanda de memoria para su almacenamiento y los requerimientos de ancho de banda para su transmisión en tiempo real. Es por ello que las técnicas de compresión adquieren gran importancia, y es en este campo precisamente donde las *wavelets* han tenido una aplicación excepcional, a tal nivel de ser el eje central del reciente estándar de compresión de imágenes digitales JPEG-2000 [3].

En muchas aplicaciones con procesamiento de imágenes usando *wavelets* se logra mejorar la calidad de la imagen y la tasa de compresión respecto a otros esquemas de procesamiento comunes basados en la transformada coseno discreta DCT (*Discrete Cosine Transform*), cuya principal desventaja es la separación de la imagen en bloques, lo cual impide eliminar la correlación existente entre las fronteras de bloques adyacentes, efecto que se hace más notable cuando la tasa de compresión es elevada. Con la *transformada wavelet discreta* DWT (*Discrete Wavelet Transform*) no hay necesidad de segmentar la imagen en bloques, permitiendo que este tipo de compresión sea más robusta a la transmisión y menos propensa a la generación de errores en el proceso de reconstrucción. Por su naturaleza multirresolución, la compresión mediante *wavelets* está especialmente indicada para aplicaciones en las que la escalabilidad y la degradación tolerable son importantes.

En los siguientes capítulos se presenta un estudio de la *teoría de wavelets* requerida para ilustrar las características de funcionamiento y desempeño de la compresión de imágenes como una de las diversas aplicaciones de esta teoría; dicho estudio está orientado al desarrollo de un algoritmo basado en la herramienta software Matlab y su toolbox "wavelet"[4], que permite comprimir y reconstruir imágenes, específicamente imágenes fijas de 512 x 512 píxeles a 256 niveles de grises empleando la familia Haar de *funciones wavelets*, familia muy usada para el análisis de imágenes que presentan cambios abruptos.

Bajo estas consideraciones, se presenta en los primeros capítulos conceptos básicos matemáticos de la *transformada wavelet* y de la compresión de imágenes, posteriormente

los aspectos tenidos en cuenta en los procesos de análisis, diseño e implementación para el desarrollo del algoritmo de simulación, y se concluye con el resultado final del proyecto, indicando la potencialidad del algoritmo desarrollado, expresando su alcance, limitaciones y en términos generales su desempeño como herramienta software de ilustración de la compresión de imágenes con *wavelets*.

2 TEORÍA BÁSICA

Tras haber visto en términos generales los principios de los *análisis de Fourier y wavelet*, y haber considerado algunas de las aplicaciones que de ellos se derivan, se procede ahora a realizar la descripción formal de las *Teorías de Fourier y wavelet*, y estudiar sus bases teóricas, realizando previamente un estudio breve de los conceptos de álgebra lineal que brindan el soporte matemático a las dos teorías.

De ningún modo se pretende restar relevancia a los diferentes elementos del álgebra lineal no tratados en este documento, pero que son fundamentos de los que sí se consideran aquí. Esta restricción de contenidos, obedece más a la suposición de que el lector posee al menos conocimientos básicos del tema y por ende es posible abordar de manera más directa las *teorías de Fourier y wavelets*, sin extenderse en el rigor matemático y dando prevalencia a la forma en que éstas se aplican.

2.1 FUNDAMENTOS EN ÁLGEBRA LINEAL

Definición 2.1. Un *espacio vectorial* está constituido de los siguientes elementos:

1. Un cuerpo K de escalares;
2. Un conjunto V de objetos llamados vectores;
3. Una operación binaria $+ : V \times V \rightarrow V$ tal que $(\boldsymbol{x},\boldsymbol{y}) \rightarrow \boldsymbol{x} + \boldsymbol{y}$, satisface:
   (a) $\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{y} + \boldsymbol{x}$
   (b) $\boldsymbol{x} + (\boldsymbol{y} + \boldsymbol{z}) = (\boldsymbol{x} + \boldsymbol{y}) + \boldsymbol{z}$
   (c) Existe un único vector $\mathbf{0} \in V$, llamado vector nulo, tal que $\boldsymbol{x} + \mathbf{0} = \mathbf{0} + \boldsymbol{x} = \boldsymbol{x}$, $\forall \boldsymbol{x} \in V$
   (d) Para cada vector $\boldsymbol{x} \in V$, existe un único vector $-\boldsymbol{x} \in V$, tal que $\boldsymbol{x} + (-\boldsymbol{x}) = \mathbf{0}$;

4. Una operación externa $\cdot : K \times V \to V$ tal que $(\alpha, \boldsymbol{x}) \to \alpha\boldsymbol{x}$, llamada multiplicación por escalar, que satisface para $\alpha, \beta \in K$ y $\boldsymbol{x}, \boldsymbol{y} \in V$:

   (a) $1 \cdot \boldsymbol{x} = \boldsymbol{x}, \forall \boldsymbol{x} \in V$

   (b) $(\alpha\,\beta) \cdot \boldsymbol{x} = \alpha\,(\beta\,\boldsymbol{x})$

   (c) $\alpha \cdot (\boldsymbol{x} + \boldsymbol{y}) = (\alpha \cdot \boldsymbol{x} + \alpha \cdot \boldsymbol{y})$

   (d) $(\alpha + \beta) \cdot \boldsymbol{x} = \alpha \cdot \boldsymbol{x} + \beta \cdot \boldsymbol{x}$

Definición 2.2. Sea *V* un espacio vectorial sobre *K*. Un *subespacio* de *V* es un subconjunto $W \subset V$ que con las operaciones de adición vectorial y multiplicación escalar sobre *V*, es él mismo un espacio vectorial.

Definición 2.3. Sea *V* un espacio vectorial sobre *K*. Una *base de V* es un conjunto de vectores linealmente independientes de *V*, que genera el espacio *V*.

Definición 2.4. Sean *V* y *W* dos espacios vectoriales sobre el cuerpo *K* notados como *V(K)* y *W(K)* respectivamente. Una función $T : V \to W$ es una *transformación lineal* si cumple que:

$$T\,(c\boldsymbol{x} + \boldsymbol{y}) = c\,(T\boldsymbol{x}) + T\boldsymbol{y}; \quad \forall \boldsymbol{x}, \boldsymbol{y} \in V \text{ y } \forall c \in K.$$

Definición 2.5. Sea $K = \mathbb{R}$ o $\mathbb{C}$ y *V*(*K*). Un *producto interno* sobre *V* es una función que asigna a cada par ordenado de vectores $\boldsymbol{x}, \boldsymbol{y}$ de *V* un escalar $< \boldsymbol{x}, \boldsymbol{y} >$ de *K*, de tal modo que $\forall \boldsymbol{x}, \boldsymbol{y} \in V$ y $\forall c \in K$ se tiene:

(a) $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$**;**

(b) $\langle cx, y \rangle = c\langle x, y \rangle$;

(c) $\langle y, x \rangle = \overline{\langle x, y \rangle}$, donde el super-rayado denota conjugación compleja;

(d) $\langle x, x \rangle > 0$ si $x \neq 0$;

obsérvese que (a), (b) y (c) implican:

(e) $\langle x, cy + z \rangle = \bar{c}\langle x, y \rangle + \langle x, z \rangle$**.**

Es claro que si $K = \mathbb{R}$ la conjugación compleja no se aplica.

Definición 2.6. La *norma* o *longitud* de *x* respecto al producto interno se define como:

$$\|x\| = \langle x, x \rangle^{1/2} \tag{2.1}$$

Definición 2.7. Sea *V* un espacio con producto interno y sean $\boldsymbol{x}, \boldsymbol{y} \in V$. Entonces $\boldsymbol{x}$ es *ortogonal* a $\boldsymbol{y}$ si $< x, y > = 0$. Ahora bien, si $S \subset V$, se dice que *S* es un *conjunto ortogonal* siempre que todos los pares de elementos distintos en *S* sean ortogonales. Además, se dice que *S* es un conjunto *ortonormal* si es ortogonal y cumple :

$$\|x\| = 1, \forall x \in S \tag{2.2}$$

Definición 2.8. Sea *H* un subespacio de *V*. El *complemento ortogonal* de *H*, denotado por $H^{\perp}$, esta dado por

$$H^{\perp} = \{x \in V : \langle x, h \rangle = 0 \, , \forall h \in H\} \tag{2.3}$$

2.2 ESPACIOS DE FUNCIONES Y ESPACIOS DE HILBERT

Sea *S* un conjunto arbitrario (, caso discreto o , caso continuo) entonces ^{S} denota el conjunto de todas las funciones $f : S \to$; en ^{S} se cumple:

a) $(k f)(s) = k f(s), \; k \in S$

b) $(f + g)(s) = f(s) + g(s)$, entonces ^{S} es espacio vectorial complejo.

Se define ahora el producto interno así,

a) Para el caso discreto donde S =

$$\langle f, g \rangle \equiv \sum_{n=-\infty}^{\infty} \overline{f(n)} \, g(n) \, , \tag{2.4}$$

con $f = g$

$$\| f \|^2 \equiv \langle f, f \rangle = \sum_{n=-\infty}^{\infty} |f(n)|^2 \tag{2.5}$$

Normalmente son sumas que divergen pero lo que en verdad interesa es el subespacio de funciones de norma finita. Es decir

$$H \equiv \left\{ f \in \mathbb{C}^{\mathbb{Z}}, \|f\|^2 < \infty \right\} \tag{2.6}$$

Nombrado como *espacio de secuencias complejas de cuadrado sumable* y es denotado por $\ell^2(\mathbf{Z})$ [2].

b) Para el caso continuo $S = \mathbb{R}$, se tiene

$$\langle f, g \rangle \equiv \int_{-\infty}^{\infty} \overline{f(t)}\, g(t) dt \tag{2.7}$$

Ahora si $f = g$,

$$\|f\|^2 \equiv \langle f, f \rangle = \int_{-\infty}^{\infty} |f(t)|^2 dt\,. \tag{2.8}$$

Pero la definición de Riemann no aplica para muchas funciones $f, g \in \mathbb{C}^{\mathbb{R}}$, lo que se resuelve con el concepto de integral de Lebesgue bajo el concepto de medida [2]. Luego si $f, g \in \mathbb{C}^{\mathbb{R}}$ son medibles lo será $\overline{f(t)}g(t)$ y $|f(t)|^2$. Luego el espacio $H = \{f : \mathbb{R} \to \mathbb{C}\}$ de funciones medibles tal que

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty \tag{2.9}$$

conforma el espacio básico de señales de variable real. En ingeniería se conocen como funciones de energía finita. Además H se conoce como el espacio de *funciones de cuadrado integrable* denotado por $\mathcal{L}^2(\mathbb{R})$ llamados *espacios de Hilbert* [2].

2.3 TEORIA DE FOURIER

La representación de fenómenos físicos puede hacerse en el dominio del tiempo $\left(f(t)\right)$ o en el dominio de la frecuencia $\left(F(w)\right)$ mediante la Transformada de Fourier, haciendo de esta una herramienta muy utilizada en aplicaciones en el campo de la ingeniería.

La representación de una señal periódica mediante la suma de funciones seno y coseno es lo que se conoce como *Serie Trigonométrica* y está descrita en la ecuación siguiente:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \{a_n \cos(nt) + b_n sen(nt)\} \tag{2.10}$$

Cuando es posible calcular todos los coeficientes a_n y b_n mediante integración de la función $f(t)$, la serie toma el nombre de *Serie de Fourier*.

$$\begin{aligned} a_0 &= \frac{1}{2\pi}\int_0^{2\pi} f(t)dt \\ a_n &= \frac{1}{\pi}\int_0^{2\pi} f(t)\cos(nt)dt \quad (n = 1, 2, \ldots) \\ b_n &= \frac{1}{\pi}\int_0^{2\pi} f(t)sen(nt)dt \quad (n = 1, 2, \ldots) \end{aligned} \tag{2.11}$$

Dado que cualquier función puede expresarse como la suma de una componente par y otra impar de $f(t)$, esto es,

$$f(t) = E(t) + O(t) \tag{2.12}$$

donde

$$\begin{aligned} E(t) &= \tfrac{1}{2}\left[f(t) + f(-t)\right] \\ O(t) &= \tfrac{1}{2}\left[f(t) - f(-t)\right] \end{aligned} \tag{2.13}$$

Así, se concluye que:

- Si f es par, $f(t) = f(-t)$, su serie de Fourier sólo contiene términos de cosenos.
- Si f es impar, $f(t) = -f(-t)$, su serie de Fourier sólo contiene términos de senos.

Ahora, la transformada de Fourier, descompone o expande una señal o función en senos y cosenos de diferentes frecuencias cuya suma corresponde a la señal original, es decir que, se es capaz de distinguir las diferentes componentes de frecuencia de la señal, y sus respectivas amplitudes. La transformada de Fourier de una función del tiempo $f(t)$ se define como

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{2.14}$$

y la transformada inversa de Fourier, como

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega \tag{2.15}$$

o la expansión de la señal original en el dominio del tiempo como

$$f(t) = \frac{1}{2\pi} \sum_{-\infty}^{\infty} F_n e^{-in\omega t} \tag{2.16}$$

Obteniéndose los coeficientes F_n de la siguiente manera: $F_n = \frac{1}{2T} \int_{t_0}^{t_0+T} f(t) e^{-in\omega t} dt$.

Así

$$f(t) = \frac{a_0}{2} + \sum \{a_n \cos(n\omega t) + b_n sen(n\omega t)\} \tag{2.17}$$

siendo

$$a_n = \text{Re}[F_n] \quad \text{y} \quad b_n = \text{Im}[F_n]. \tag{2.18}$$

Hay que tener en cuenta que la transformada de Fourier maneja propiedades como el escalamiento y translación (en tiempo y en frecuencia), convolución[1], correlación y un importante teorema conocido como *Teorema de Parseval* el cual establece que la energía de la señal es siempre la misma sin importar si se analiza en el dominio del tiempo o de la frecuencia [2].

$$Energía\ Total = \int_{-\infty}^{\infty} |f(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(w)|^2 dw \tag{2.19}$$

Para el caso práctico se requiere implementar la transformada dentro de un ambiente computacional y es necesario trabajar sólo con valores discretos de *f*(*t*), ya que los computadores no manejan funciones continuas sino muestras que conforman la señal, es decir, valores de la forma f_k con $k = 0, 1, 2, \ldots$. Esto significa que mediante el uso de un computador es posible calcular la transformada $F(w)$ sólo para valores discretos de *w*, es decir, se obtiene valores de la transformada de la forma F_n con $n = 0, 1, 2, \ldots$

---

[1] $f(t) * h(t) = \int_{-\infty}^{\infty} f(\tau) h(t-\tau) d\tau$

$f(t) * h(t) \Leftrightarrow F(w) H(w)$

Si se supone ahora que $f(t)$ es una señal periódica de período T y que sólo se conoce sus valores en N instantes igualmente espaciados, entonces, si $f(kT_s)$ corresponde a la k–ésima muestra de $f(t)$ y $F(nw_s)$, donde $w_s = 2\pi f_s$ (f_s es la frecuencia con la que se realiza el muestreo) corresponde a la n-ésima muestra de $F(w)$, y además si se define a N como el número de muestras de la señal o longitud de la señal, se obtiene la forma discreta de la *Transforma Fourier* así,

$$F_n = \sum_{k=0}^{N-1} f_k e^{-\frac{i2\pi kn}{N}} \qquad \text{con } n, k = 0, 1, 2, \ldots, N\text{-}1. \tag{2.20}$$

Ahora, como

$$f_k = f(k\frac{T}{N}) \qquad \text{con } T_s = \frac{T}{N} \tag{2.21}$$

entonces

$$F_n = F(nw_s) \qquad \text{con } w_s = \frac{2\pi}{T} \tag{2.22}$$

Siempre que F_n tenga periodo N al igual que f_k , al conjunto de coeficientes (F_n) $n = 0, 1, 2, \ldots, N\text{-}1$ se le conoce como la *Transformada Discreta de Fourier* o DFT de las muestras (f_k), $k = 0, 1, 2, \ldots, N\text{-}1$.

De forma similar la *Inversa de la Transformada Discreta de Fourier* o IDFT, está definida mediante la siguiente expresión:

$$f_k = \frac{1}{N}\sum_{n=0}^{N-1} F_n e^{\frac{i2\pi kn}{N}} \qquad k = 0, 1, 2, \ldots, N\text{-}1 \tag{2.23}$$

La transformada de Fourier permite conocer las componentes de frecuencia existentes en la señal y sus respectivos aportes energéticos, es decir, presenta una perfecta resolución en frecuencia que la hace una herramienta muy útil para el análisis de señales estacionarias. Pero tiene el inconveniente de no permitir conocer de manera precisa cuándo o dónde las diferentes componentes de frecuencia se encuentran en la señal como es el caso de señales cuasi-estacionarias o no estacionarias cuyo contenido espectral varía con el tiempo. En otras palabras, la transformada de Fourier posee una muy pobre resolución en tiempo.

Denis Gabor en 1946 intenta proporcionar resolución en el tiempo utilizando un procedimiento llamado *ventaneo* que consiste en dividir una señal $x(t)$ en pequeños segmentos a través del tiempo de tal manera que se pueda asumir que la señal es estacionaria en cada segmento y así calcular la transformada de Fourier clásica para cada porción de la señal.

La señal se divide mediante una *función tiempo – ventana h(t)* cuyo ancho o soporte corresponde a la longitud de cada segmentación de la señal. Esta función encuadra la señal alrededor de un instante de tiempo τ y permite calcular su transformada de Fourier, luego se traslada la función ventana para cubrir una nueva porción de la señal sin que se traslapen con la cubierta anteriormente analizada y se vuelve a calcular su transformada de Fourier. Este proceso se repite hasta cubrir la totalidad de la señal.

El resultado anterior se conoce como la *Transformada Corta de Fourier* o STFT y se define en forma matemática de la siguiente manera:

$$STFT(t,w) = \int_{-\infty}^{\infty} x(t)h^*(\tau - t)e^{-iwt}dt \qquad (2.23)$$

Ahora si se considera a $h(t)$ como una función ventana de valores solo reales de tal manera que $h(-t) = h^*(t)$ entonces nos queda

$$STFT(t,\xi) = \int_{-\infty}^{\infty} x(t)h(\tau - t)e^{-i\xi t}dt \qquad (2.24)$$

Ahora bien, el soporte de la ventana constituye un parámetro de gran importancia ya que establece un compromiso entre resolución en tiempo y resolución en frecuencia, de tal manera que al establecer una ventana angosta se analiza una pequeña parte de la señal teniéndose una buena resolución en tiempo pero pobre en frecuencia ya que solo se conoce una pequeña fracción del espectro total de la señal. Por el contrario, si la ventana en muy ancha se tiene una buena resolución en frecuencia pero pobre resolución en tiempo. En el caso extremo una ventana de ancho infinito corresponde a la transformada de Fourier clásica. Por lo tanto un defecto de la STFT es que no puede entregar una buena resolución tanto en tiempo como en frecuencia de manera simultánea ya que el soporte de la ventana es fijo.

A partir de este inconveniente surge la inquietud sobre si es posible tener una ventana con rango dinámico que permita realizar un análisis de la señal de manera ideal, es decir, tener una buena resolución en tiempo para frecuencias altas, y una buena resolución en frecuencia para analizar las frecuencias bajas. Precisamente como solución a este problema tiene lugar la herramienta matemática llamada *transformada wavelet*.

2.4 *TRANSFORMADA WAVELET*

De manera similar como la transformada de Fourier descompone una función o señal a analizar en funciones senoidales, la *transformada wavelet* la descompone en términos de funciones denominadas *wavelets*. Esta transformada comprende básicamente la *transformada continua wavelet* (CWT) y la t*ransformada discreta wavelet* (DWT), herramientas matemáticas que posibilitan el análisis de señales proporcionando información en el dominio del tiempo y en el dominio de la frecuencia. El análisis mediante la CWT se efectúa sobre señales analógicas de energía finita [2] (definida en el teorema de Parseval). Sin embargo, el procesamiento digital de señales, tal como su nombre lo indica, se aplica sobre muestras de datos digitales en tiempos discretos [5].

2.4.1 *Transformada continua wavelet* (CWT)

Así como lo hace la STFT, esta transformada utiliza una función ventana con la cual se analiza por intervalos una señal, sin embargo, esta ventana es de ancho variable. La CWT intenta expresar una señal $x(t)$ continua en el tiempo, mediante una expansión de términos o coeficientes proporcionales al producto interno entre la señal y diferentes versiones escaladas y trasladadas de una función prototipo $\psi(t)$, conocida como *wavelet madre*. Entre este tipo de *wavelets* una de las más conocidas es la *Haar* representada en la figura 1, siendo ésta la más antigua y simple de todas, no obstante, debido a su forma es muy utilizada para analizar señales que presentan cambios abruptos.

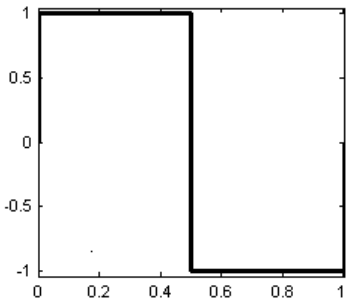

Figura 1. ***Función Wavelet Haar***

Algunos ejemplos de otras *wavelets madre* se muestran en la figura 2.

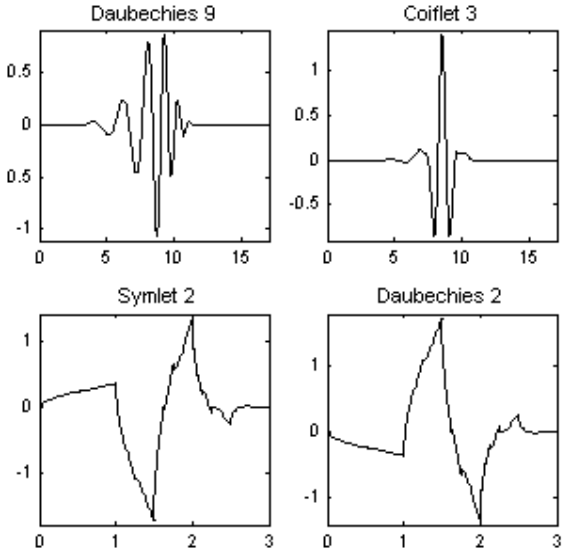

Figura 2. ***Funciones wavelets*** **más comunes**

Partiendo de que tanto la señal $x(t)$ como la *función wavelet madre* $\psi(t)$ son de energía finita [2], entonces se define la CWT de una señal $x(t)$ como:

$$\text{CWT}\,(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t)\,\psi\left(\frac{t-b}{a}\right) dt\,, \quad a, b \in \mathbb{R} \quad \text{y} \quad a \neq 0 \tag{2.25}$$

donde la variable a determina el ancho o soporte de la función ventana y la variable b determina la ubicación en el dominio del tiempo de la misma. De acuerdo a esto se identifica a la variable a como *variable de escala*, la cual permite comprimir (reducir la

escala) o dilatar (subir la escala) la función $\psi(t)$, en otras palabras, establece el grado de resolución con el cual se analiza la señal, mientras que la variable b se llama *variable de traslación* [6].

De acuerdo a la definición de la CWT puede concluirse que más que una representación tiempo-frecuencia, es una representación tiempo-escala, en tanto que la información global de la señal, es decir, las bajas frecuencias, se analizan con altas escalas y se obtiene buena resolución en frecuencia, mientras que los detalles de la señal, correspondientes a las altas frecuencias, se analizan con bajas escalas, con lo que se obtiene buena resolución en tiempo. En este sentido, se dice que la escala y la frecuencia tienen una relación inversamente proporcional. La representación gráfica de la CWT se realiza en un plano denominado *plano tiempo-escala*, mostrado en la figura 3.

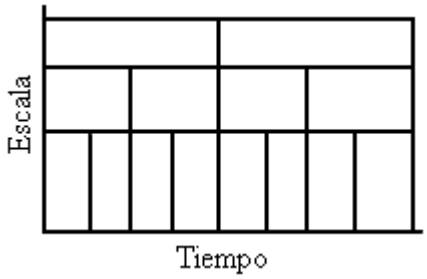

Figura 3. Plano tiempo -escala en la CWT

Desde un punto de vista intuitivo, la CWT consiste en calcular un índice de semejanza entre la señal que está siendo analizada, y la w*avelet*, tal como se muestra en la figura 4. El proceso de cálculo de la CWT se puede describir mediante los siguientes pasos [7]:

1. Tomar una *wavelet madre* y compararla con un intervalo al principio de la señal original.
2. Dados dos valores a y b, calcular un coeficiente $c_{a,b}$ (lo cual se estudiará en la sección 2.4.4) que representa la correlación entre la w*avelet* y la sección de la señal bajo análisis. Entre mayor sea el valor del coeficiente, mayor es la similitud, de ahí que los resultados dependen de la *wavelet* elegida para hacer dicho análisis.

3. Desplazar la w*avelet* en el sentido positivo del eje temporal, y repetir los pasos anteriores hasta que se haya cubierto la totalidad de la señal.
4. Escalar la w*avelet* en el tiempo y repetir los pasos 1 a 3.

Un ejemplo de este proceso se representa en la figura 4.

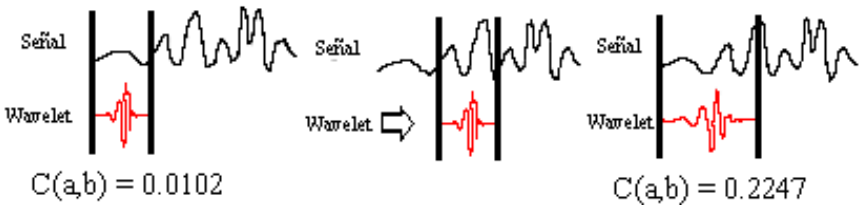

Figura 4. Modo de funcionamiento de la CWT

La definición de la CWT puede expresarse en términos de la transformada de Fourier aplicando el Teorema de Parseval (estudiado en la sección 2.3), como:

$$\text{CWT}\,(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} X(\omega)\Psi(a\omega)e^{2\pi i\omega b}d\omega \qquad (2.26)$$

donde $X(\omega)$ y $\Psi(\omega)$ son las transformadas de Fourier de $x(t)$ y $\psi(t)$ respectivamente. Es importante anotar que para que el *análisis wavelet* de una señal sea posible y se logre una perfecta reconstrucción de la misma a partir de la transformada, la función $\psi(t)$ debe cumplir con la condición de admisibilidad [2], la cual implica que su valor medio debe ser igual a cero, es decir:

$$\Psi(0) = 0 \qquad (2.27)$$

El cumplimiento de esta condición significa que ψ debe tener valores tanto positivos como negativos, en otras palabras, se requiere que sea una onda. Además, teniendo en cuenta que la *función wavelet* analiza la señal por intervalos de tiempo de acuerdo al tamaño de la ventana, se dice que ésta es de *soporte compacto*, es decir, es una onda definida sobre un intervalo de tiempo finito, de ahí su nombre *wavelet* u ondita.

2.4.2 *Transformada discreta wavelet* (DWT)

A nivel teórico se habla de la *transformada wavelet continua*, donde dicha continuidad radica en el hecho que la variable de escala y la variable de traslación varían en forma continua. Sin embargo a nivel práctico, es decir, a nivel computacional, se hace necesario discretizar la transformada, y la solución más lógica es asignar valores discretos a las dos variables mencionadas, dando lugar así a la *transformada wavelet discreta* DWT. La forma más común de discretizar los valores de *a* y *b* es utilizar una red diádica [5], es decir, $a = 2^{-j}$ y $b = k\,2^{-j}$, de tal manera que el conjunto de *funciones wavelets* determinadas por:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right); \quad a,\, b \in \mathbb{R},\, a \neq 0 \tag{2.28}$$

adquiere la forma:

$$\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi\,(2^{j}t - k); \quad j,\, k \in \mathbb{Z}. \tag{2.29}$$

Este conjunto de funciones se conoce como la *versión diádica discretizada* de la *función wavelet* o como *conjunto de expansión wavelet* [5], y es el resultado de traslaciones y escalamientos de la *función wavelet madre* $\psi(t)$. El factor $2^{\frac{j}{2}}$ se denomina *constante de normalización* y se hace necesario para cumplir con la condición de ortonormalidad (expuesta en la sección 2.1).

De otro lado, la *función wavelet madre* $\psi(t)$ lleva asociada consigo una función escala $\phi(t)$, de tal manera que con una de estas funciones o con ambas es posible aproximar cualquier función $f(t) \in \mathcal{L}^2(\mathbb{R})$, de la siguiente forma:

$$f(t) = \sum_k \sum_j c_{j,k}\ \phi(t) + \sum_k \sum_j d_{j,k}\ \psi(t); \ j,k \in \mathbb{Z} \tag{2.30}$$

En la expresión (2.30) se ha aproximado *f* en términos de la función escala $\phi(t)$ y de la *función wavelet* $\psi(t)$. A continuación se estudian más a fondo estos dos tipos de funciones.

2.4.2.1 Función escala $\phi(t)$

Con el propósito de tratar el concepto de multirresolución más adelante, se requiere definir la función escala y a partir de ésta, definir la *función wavelet*. Una función escala en términos de desplazamientos de una función escala básica $\phi(t)$ se define como:

$$\phi_k(t) = \phi(t-k); \quad k \in \mathbb{Z} \tag{2.31}$$

donde $\phi \in \mathcal{L}^2(\mathbb{R})$ se define como la unidad en el intervalo $[0,1)$ y cero en el resto (esta función escala básica se estudia con mayor extensión en el anexo A). Ahora bien, con la expresión para $\phi_k(t)$ se genera el subespacio dado en la relación (2.32)

$$V_0 = \overline{Span_{k\in\mathbb{Z}}\{\phi(t)\}}. \tag{2.32}$$

De otro lado, es posible incrementar el tamaño del subespacio cambiando la escala de la función, así, escalando y desplazando en tiempo la función escala tal como se hizo con la función $\psi_{j,k}(t)$, se genera una familia de funciones $\{\phi_{j,k}(t) \mid j,k \in \mathbb{Z}\}$ en dos dimensiones (versión diádica discretizada [5]) como sigue:

$$\phi_{j,k}(t) = 2^{\frac{j}{2}}\phi(2^j t - k); \quad j, k \in \mathbb{Z}, \tag{2.33}$$

la cual define un subespacio $V_j \subset \mathcal{L}^2(\mathbb{R})$ para cada $j \in \mathbb{Z}$ de la siguiente manera:

$$V_j = \overline{Span_{k\in\mathbb{Z}}\{\phi_{j,k}(t)\}}. \tag{2.34}$$

Entonces una función $f(t)$ está en V_j si puede expresarse como:

$$f(t) = \sum_{k\in\mathbb{Z}} c_{j,k}\ \phi_{j,k}(t) \tag{2.35}$$

con coeficientes $c_{j,k}$ obtenidos mediante el producto interno entre la función $f(t)$ y la función escala así:

$$c_{j,k} = \left\langle f(t)\ ,\ \phi_{j,k}(t) \right\rangle = \int_{-\infty}^{\infty} \left| f(t)\ \ \phi_{j,k}(t) \right| dt. \qquad (2.36)$$

Entre mayor sea el valor de j, mayor es la expansión puesto que $\phi_{j,k}(t)$ es más estrecha y se desplaza en pasos más pequeños, con ello entonces se representa los detalles de la función $f(t)$. Por otro lado, entre más pequeño sea el valor de j, $\phi_{j,k}(t)$ es más alargada y se desplaza en pasos grandes, por lo cual representa información general de $f(t)$.

Algunas características de la función escala son las siguientes [5]:

1. Para cada $j \in \mathbb{Z}$, $\{ \phi_{j,k} \mid k \in \mathbb{Z} \}$ forma una base ortonormal para el subespacio $V_j \subset \mathcal{L}^2(\mathbb{R})$.
2. Los subespacios V_j están anidados, es decir: $\forall\, j \in \mathbb{Z}$, $V_j \subset V_{j+1}$, de tal manera que los subespacios V_j incluyen más funciones de $\mathcal{L}^2(\mathbb{R})$ a medida que j crece, (ver figura 5).
3. La función escala $\phi(t)$ tiene soporte compacto, es decir que existe un subconjunto del dominio de $\phi(t)$ donde ésta no es cero. En términos matemáticos esta propiedad indica que:

$$sup\ \{\phi\} = \{\ x \in \mathbb{R}\ \mid \phi(x) \neq 0\ \}. \qquad (2.37)$$

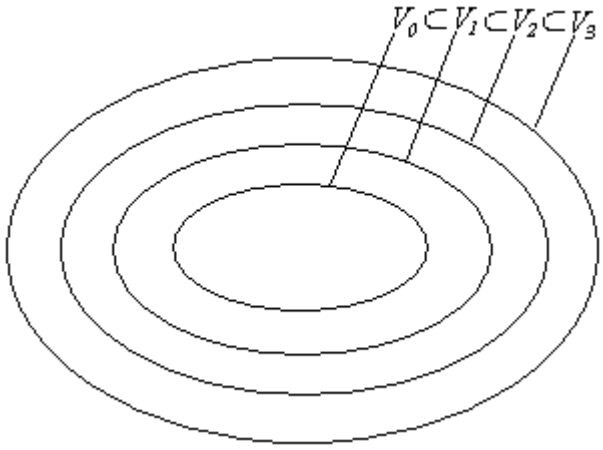

Figura 5. Espacios anidados generados por la función escala

De acuerdo a la propiedad 2 se tiene que si $f(t) \in V_j \Rightarrow f(t) \in V_{j+1}$ lo cual implica que existen unos coeficientes que permiten que la función $f(t)$ pueda ser expresada en términos de una suma de $f(2t)$ ajustada y trasladada. En este sentido la función escala puede expresarse como:

$$\phi(t) = \sum_{k \in \mathbb{Z}} \sqrt{2}\, h(k)\, \phi(2t - k) \tag{2.38}$$

Comúnmente esta ecuación es conocida como *ecuación básica de recursión* o *ecuación de escala*, la cual surge de la expresión (2.33) con $j = 1$. Los coeficientes $h(k)$ se llaman *coeficientes de escala* y pueden ser reales o complejos, los cuales se hallan realizando el producto interno entre $\phi(t)$ es decir (2.38) y $\sqrt{2}\phi(2t - k)$ (más adelante en la sección 2.4.3 se calculan estos coeficientes para el caso de $k = 0$ y $k = 1$), mientras que el factor $\sqrt{2}$ mantiene normalizada a la función en el subespacio V_1 .

2.4.2.2 *Función wavelet* $\psi(t)$

La mejor manera de parametrizar o describir las características importantes de una función o señal no es incrementar el valor de j para aumentar el tamaño del subespacio generado por la función escala, sino más bien considerar un conjunto de funciones que genere las diferencias entre los espacios generados por las diferentes escalas de la función escala. Dichas funciones son las *wavelets*, que generan el subespacio:

$$W_j = \overline{Span_{\,k \in \mathbb{Z}}\, \{\psi_{j,k}(t)\}} \tag{2.39}$$

que corresponde al complemento ortogonal de V_j en V_{j+1}, es decir $V_{j+1} = V_j \oplus W_j$, por ende todos los componentes de V_j son ortogonales a todos los componentes de W_j. Se requiere entonces que:

$$\left\langle \phi_{j,l}(t)\ ,\ \psi_{j,k}(t) \right\rangle = \int_{-\infty}^{\infty} \overline{\phi_{j,l}(t)}\ \ \psi_{j,k}(t)\ dt = 0;\ \ \forall\ j,\ k,\ l\ \in \mathbb{Z} \qquad (2.40)$$

con

$$\psi_{j,k}(t) = 2^{\frac{j}{2}} \psi(2^{j}t - k);\quad j,\ k \in \mathbb{Z}. \qquad (2.41)$$

De esta manera, cualquier función $f(t) \in W_j$ puede ser representada como:

$$f(t) = \sum_{k \in \mathbb{Z}} d_{j,k}\ \psi_{j,k}(t) \qquad (2.42)$$

donde los coeficientes $d_{j,k}$ se calculan mediante el producto interno entre la señal analizada $f(t)$ y la *función wavelet*:

$$d_{j,k} = \left\langle f(t), \psi_{j,k}(t) \right\rangle = \int_{-\infty}^{\infty} \left| f(t)\ \psi_{j,k}(t) \right| dt \qquad (2.43)$$

La *función wavelet básica* o *wavelet madre* $\psi(t)$ resulta de (2.41) para $j = 0$ y $k = 0$. Entonces $\psi(t) = \psi_{0,0}(t) \in W_0$ y como $V_1 = V_0 \oplus W_0$ se tiene que $\psi(t) \in V_1$. De este modo, es posible representar la función $\psi(t)$ de manera análoga como se hizo con la función escala, tal como sigue:

$$\psi(t) = \sum_{k \in \mathbb{Z}} \sqrt{2}\ h_1(\mathrm{k})\ \phi(2t - k) \qquad (2.44)$$

donde los coeficientes $h_1(k)$ se llaman *coeficientes wavelet* y se hallan realizando el producto interno entre $\psi(t)$ es decir (2.44) y $\sqrt{2}\phi(2t - k)$ (en el anexo A se muestra el procedimiento matemático para los casos *k = 0* y *k = 1*).

Finalmente, a partir de las relaciones obtenidas en (2.36) y (2.43) puede aproximarse cualquier función $f(t) \in \mathcal{L}^2(\mathbb{R})$ de acuerdo a la ecuación (2.30).

2.4.3 Principios de multirresolución

El análisis multirresolución consiste básicamente en aproximar una función $f(t)$ en distintos niveles de resolución $\{f_1(t),\ f_2(t)\ \ f_3(t),\ ...\}$, lo cual proporciona una descomposición multiescala [8] de la forma:

$$f(t) = f_0(t) + \sum_{j\geq 0} g_j(t) \tag{2.45}$$

donde cada g_i está dado por $g_j(t) = f_{j+1}(t) - f_j(t)$ y representa el error cometido al aproximar $f_{j+1}(t)$ mediante $f_j(t)$, es decir, la fluctuación entre dos niveles sucesivos de resolución.

Para efecto del análisis multirresolución se utiliza una función $\psi(t)$ cuidadosamente escogida según la señal a analizar, la cual está bien localizada tanto en tiempo como en frecuencia, además, sus translaciones y escalamientos generan una base $\{\psi_{j,k}(t) | \ j,k \in \mathbb{Z}\}$ que expande $g_j(t)$ como se muestra en la siguiente relación:

$$g_j(t) = \sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(t) \tag{2.46}$$

En la anterior relación, los $d_{j,k}$ son coeficientes escalares llamados *coeficientes wavelet* que fueron expresados en la ecuación (2.43).

Un análisis multirresolución requiere un anidamiento de los espacios generados por las funciones escala, así:

$$\ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots \subset \mathcal{L}^2 \tag{2.47}$$

en forma general se tiene:

$$V_j \subset V_{j+1} \qquad \forall j \in \mathbb{Z} \tag{2.48}$$

con

$$V_{-\infty} = \{0\}, \quad V_{\infty} = \mathcal{L}^2 \tag{2.49}$$

Lo anterior indica que el espacio que contiene las señales de más alta resolución contiene también las de más baja resolución. Por otra parte debido a la definición de V_j, estos espacios cumplen con la siguiente condición de escalamiento:

$$f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1} \tag{2.50}$$

lo anterior implica que los elementos pertenecientes a un espacio son solo versiones escaladas de los elementos del siguiente espacio. Para explicar mejor esta propiedad, se representa a continuación la función escala de *Haar* (estudiada más a fondo en el anexo A) mediante versiones escaladas y trasladadas de ella misma en el intervalo $[0,1)$. Esto es:

$$\phi_{0,0}(t) = \phi_{1,0}(t) + \phi_{1,1}(t) \tag{2.51}$$

donde $\phi_{1,0}$ está definida en el intervalo $[0,1/2)$ y $\phi_{1,1}$ en el intervalo $[1/2,1)$.

Aplicando la ecuación de recursión (2.38), se obtiene otra forma de ver la relación (2.51) así:

$$\phi(t) = h(0)\sqrt{2}\phi(2t) + h(1)\sqrt{2}\phi(2t-1) \tag{2.52}$$

donde

$$h(0) = \int_0^{\frac{1}{2}} \phi(t)\sqrt{2}\phi(2t)dt \quad \Rightarrow \quad h(0) = \frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}} \tag{2.53}$$

$$h(1) = \int_{\frac{1}{2}}^{1} \phi(t)\sqrt{2}\phi(2t-1)dt \quad \Rightarrow \quad h(1) = \frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}} \tag{2.54}$$

entonces (2.52) queda como

$$\phi(t) = \phi(2t) + \phi(2t-1) \tag{2.55}$$

mostrando que la función escala se puede representar por una versión de ella misma escalada más otra versión de ella misma escalada y trasladada.

En la figura 5, se mostró la relación entre los espacios expandidos por las funciones escala en sus distintos niveles de resolución, llegando a la conclusión nuevamente que la familia

de funciones $\phi_{j,k}(t)$ generan el espacio V_j expresado en la ecuación (2.34). Ahora bien, en la figura 6 se muestra en forma esquemática lo explicado anteriormente para la función *sen(t)*.

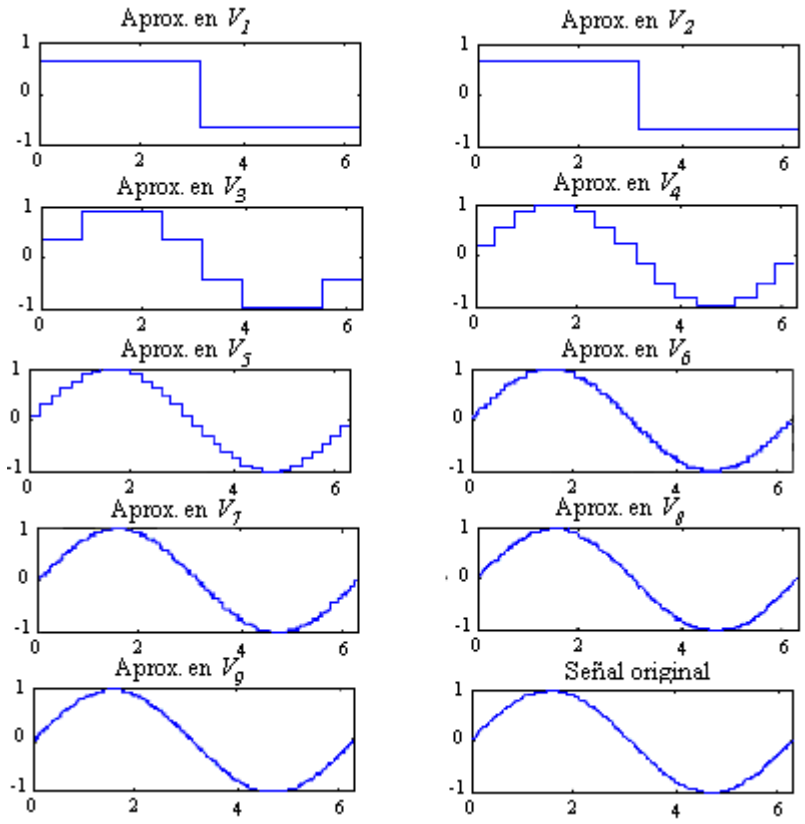

Figura 6. Representación de la función *sen(t)* por la función escala en distintos espacios

Sin embargo, y como se mencionó en la sección 2.4.2 para obtener una mejor representación de la señal, lo que se hace es definir un nuevo espacio W_j, llamado *espacio wavelet*, como el complemento ortogonal de V_j en V_{j+1}, así:

$$V_j \oplus W_j = V_{j+1}. \tag{2.56}$$

Una descripción gráfica de la anterior relación se ilustra en la figura 7, mostrando cómo los *espacios wavelet* residen en los espacios de escalamiento, fácilmente se puede observar que los espacios W_2, V_2, V_1 y V_0 se encuentran contenidos en V_3, o dicho de otra forma, V_3 está

conformado por V_0, W_0, W_1 y W_2. De acuerdo a esto, el espacio V_3 puede ser representado de la forma:

$$V_3 = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \tag{2.57}$$

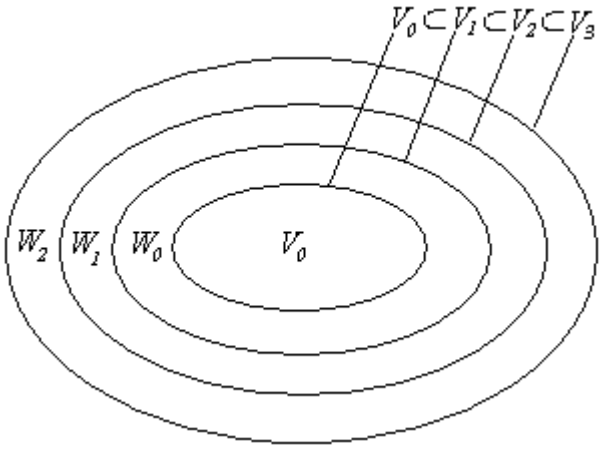

Figura 7. ***Espacios wavelets***

La anterior relación se puede generalizar a todos los espacios siguientes, así:

$$\mathcal{L}^2 = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \ldots \tag{2.58}$$

donde V_0 es el espacio inicial, expandido por la función escala $\phi(t\text{-}k)$, y los espacios W_j entregan información más detallada de la señal a medida que *j* crece. Ahora bien, la escala que se utilice para expandir el espacio inicial es una decisión personal, lo cual depende básicamente del análisis que se realice y de la señal en cuestión. Por lo tanto, se puede representar el espacio $\mathcal{L}^2$ partiendo de una resolución más alta, por ejemplo $j = 8$ con lo que resulta:

$$\mathcal{L}^2 = V_8 \oplus W_8 \oplus W_9 \oplus \ldots \tag{2.59}$$

De esta manera, se puede tomar una escala negativa para el espacio inicial, como por ejemplo $j = -\infty$ con lo cual se tiene:

$$\mathcal{L}^2 = \ldots \oplus W_{-2} \oplus W_{-1} \oplus W_0 \oplus W_1 \oplus \ldots \tag{2.60}$$

En la anterior ecuación, se representó el espacio $\mathcal{L}^2$ solo con *espacios wavelets,* que conlleva a la siguiente ecuación:

$$V_0 = W_{-\infty} \oplus \ldots \oplus W_{-1} \tag{2.61}$$

mostrando de nuevo que se puede escoger cualquier resolución para el espacio inicial.
En la figura 8 se muestra la proyección de una función en diferentes *espacios wavelet*, además se observa que a partir del espacio W_3 los *coeficientes wavelet* se concentran en puntos donde la función tiene pendiente distinta de cero. Lo anterior se debe a que las *wavelets* detectan los cambios de la función, así por ejemplo para el caso de la *wavelet Haar*, el detalle se obtiene de la resta de dos muestras sucesivas de la función discreta, lo cual se analizará más adelante, implicando lo anterior que los puntos donde la pendiente es más suave, la diferencia tenderá a cero, junto con la amplitud del *coeficiente wavelet*.

Ahora bien, la función que expande el espacio W_j es la *wavelet madre* $\psi_{j,k}(t)$. Además como $W_0 \subset V_1$, la *función wavelet* $\psi(t)$ se puede representar por una suma de funciones escala, escaladas y trasladadas, lo cual se representó en la ecuación (2.44), siendo lo anterior también válido para la función $\phi(t)$, ya que $V_0 \subset V_1$, por lo que esta función escala puede representarse como se mostró en la ecuación básica de recursión (2.38) de la sección 2.4.2.

Por otro lado, una función pertenece al espacio W_j sí se puede representar por medio de la función prototipo de una *wavelet madre* de la forma:

$$\psi_{j,k}(t) = 2^{\frac{j}{2}} \psi\left(2^j t - k\right) \tag{2.62}$$

donde 2^j es la escala de *t*, además *k* es la traslación en *t* y $2^{\frac{j}{2}}$ mantiene constante la norma de la *wavelet* en diferentes escalas. Como se observa, la variable *j* representa el *espacio wavelet* en el cual está trabajando la función madre, análogamente es la misma situación para la función escala.

Al igual que la función escala *Haar*, la cual se representó como versiones escaladas y trasladadas de ella misma, la *función wavelet Haar* también puede ser representada en el

intervalo $[0,1]$ como una combinación lineal de las funciones de escalamiento que expanden el espacio V_1, dicho procedimiento matemático se muestra en el anexo A.

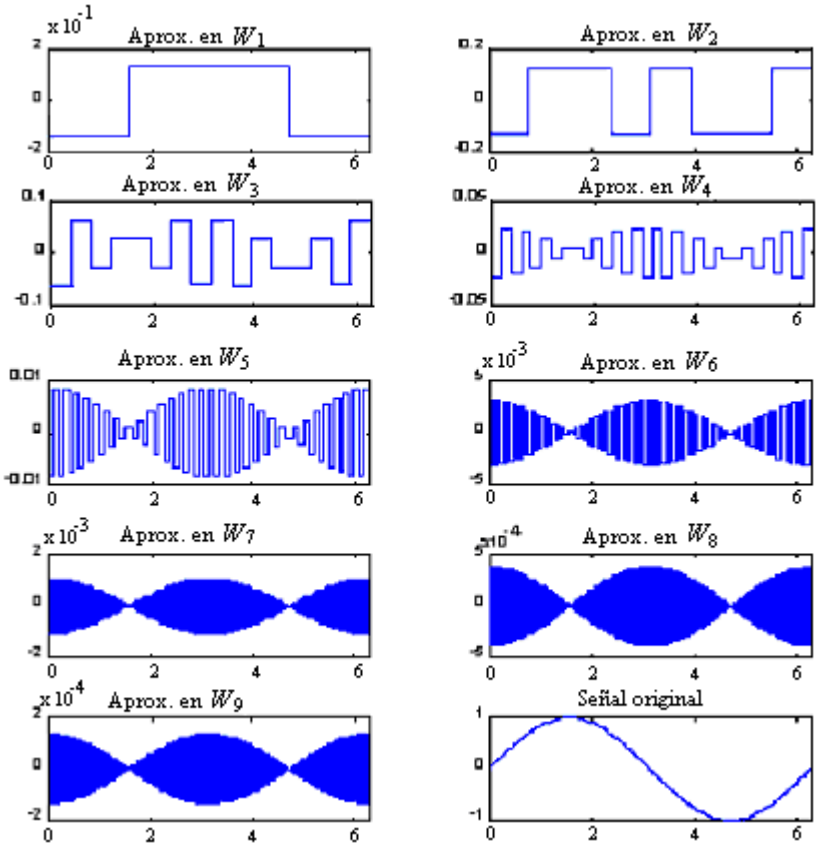

Figura 8. Proyección de una función en diferentes *espacios wavelet*, usando el sistema Haar

A continuación, en la ecuación (2.63) se muestra la expresión que representa la *transformada discreta wavelet* (DWT), que es la misma ecuación (2.30), con la diferencia que ahora se involucra la forma en que corren las sumatorias dadas con el propósito de poder realizar una descomposición eficiente de una señal o función:

$$f(t) = \sum_{k=0}^{2^{j0}-1} c_{j_0,k}\phi_{j_0,k}(t) + \sum_{j=j_0}^{N-1}\sum_{k=0}^{2^j-1} d_{j,k}\psi_{j,k}(t) \quad j,k \in \mathbb{Z}^+. \tag{2.63}$$

En la anterior ecuación, los $c_{j_0,k}$ son los coeficientes de escala, los $d_{j,k}$ son los *coeficientes wavelet* y j_0 define el espacio inicial V_{j_0} que es el de menor resolución. Ahora bien, dependiendo de j_0 es que el resto de los índices irán corriendo, además, 2^N es la longitud de

la señal *f(n)*. Esta longitud limita el nivel de descomposición de una señal, ya que no tiene sentido representarla en un espacio V_{j_n} si ya se encuentra en él.

El gran inconveniente de desarrollar la DWT es el gran tamaño de los cálculos matemáticos necesarios en cada operación, por ello es necesario buscar una manera eficiente de realizar esta DWT, la cual conduce a conocer la teoría de banco de filtros que a su vez conduce a la obtención de la *transformada rápida wavelet* (FWT).

2.4.4 *Transformada rápida wavelet* (FWT) y banco de filtros

Sea una señal $f(t) \in \mathcal{L}^2$ conocida para todo *t* (o para una discretización en el dominio del tiempo lo suficientemente densa), de acuerdo con la propiedad del análisis multirresolución estudiada en la sección 2.4.3, el espacio que contiene las señales de más alta resolución contiene también las de más baja resolución (ver ecuación (2.49)), lo que implica que $V_\infty = \mathcal{L}^2(\square)$, en otras palabras, $\mathcal{L}^2$ está formado por la unión de todos los V_j, lo anterior se representa en la siguiente expresión:

$$\overline{(U_{j\in\square} V_j)} = \mathcal{L}^2(\square) \tag{2.64}$$

por lo tanto se puede aproximar *f* tanto como se desee mediante un modelo $f_n \in V_n$ donde $n \in \square$, de modo que este modelo se puede representar como una combinación lineal de funciones escala así:

$$f_n(t) = \sum_{k\in\square} c_{n,k}\, 2^{\frac{n}{2}} \phi(2^n t - k) \tag{2.65}$$

donde los $c_{n,k}$ son coeficientes escalares por medio de los cuales se representa la señal en el dominio discreto o digital.

La relación dada en (2.65), cobra importancia en la medida en que los algoritmos diseñados para la DWT en la representación de una señal mediante funciones escala, se apliquen a datos de entrada que han sido modelados mediante una función escala, es decir, sobre el conjunto de coeficientes dado en la expresión (2.66).

$$\{c_{n,k}; k \in \mathbb{Z}\}. \tag{2.66}$$

Observe que en el conjunto anotado en la anterior expresión, solo varía *k*, ya que *n* se utiliza para denotar el modelo f_n respectivo que se utiliza para aproximar la señal original *f*.

La interpolación es uno de los métodos más efectivos para realizar el proceso de modelamiento explicado anteriormente [9], consiste en escoger los $c_{n,k}$ de tal forma que $f_n(t)$ concuerde con la representación discreta $f\left(\frac{k}{2^n}\right)$ para $t = \frac{k}{2^n}$, esto es:

$$f_n\left(\tfrac{k}{2^n}\right) = f\left(\tfrac{k}{2^n}\right) \quad k \in \mathbb{Z} \tag{2.67}$$

La relación (2.67) es la llamada *representación diádica* de una señal [5], puesto que el intervalo de tiempo dado para cada muestreo está determinado por una potencia de 2. Lo anterior significa que un modelo f_n de una señal analógica *f* corresponde a un conjunto de valores discretos con longitud $N = 2^n$.

2.4.4.1 Descomposición de señales unidimensionales (Análisis)

En la sección anterior se expuso la forma de aproximar una función *f* por medio de un modelo $f_n \in V_n$, donde si f_n es una representación de la señal original solo mediante funciones escala, es posible escribirla de la siguiente manera:

$$f_n = f_{n-1} + g_{n-1} \tag{2.68}$$

La expresión (2.68) es una implicación directa de las relaciones (2.69), (2.70) y (2.71), (conceptos que fueron analizados en las secciones 2.4.2 y 2.4.3):

$$V_{n-1} \subset V_n \quad \wedge \quad W_{n-1} \subset V_n \tag{2.69}$$

$$V_n = V_{n-1} \oplus W_{n-1} \tag{2.70}$$

$$f_{n-1} \in V_{n-1} \quad \wedge \quad g_{n-1} \in W_{n-1} \tag{2.71}$$

Ahora bien, el desarrollo de bancos de filtros y de los algoritmos diseñados para la DWT, no se relaciona directamente con las funciones escala y *wavelet*, sino con los coeficientes relacionados a estas funciones, por lo tanto, el primer paso en la descomposición es poder encontrar los coeficientes $c_{n-1,k}$ y $d_{n-1,k}$ en términos de $c_{n,k}$.

Con este propósito, de (2.65) y (2.67), se puede representar una señal unidimensional de energía finita [2] mediante los coeficientes $c_{n,k}$ como:

$$f(t) = \sum_{k \in \mathbb{Z}} c_{n,k} 2^{\frac{n}{2}} \phi(2^n t - k). \tag{2.72}$$

Se sabe también que tanto la función escala $\phi(t)$ como la *wavelet* $\psi(t)$, generan bases ortogonales en $\mathcal{L}^2$, así que el cálculo de $c_{n-1,k}$ y $d_{n-1,k}$ se realiza a través del producto interno de la señal con la función *escala* y *wavelet* respectivamente, esto es:

$$c_{n-1,k} = < f(t), \phi_{n-1,k}(t) > = \int_{-\infty}^{\infty} f(t) 2^{\frac{n-1}{2}} \phi(2^{n-1} t - k) dt \tag{2.73}$$

$$d_{n-1,k} = < f(t), \psi_{n-1,k}(t) > = \int_{-\infty}^{\infty} f(t) 2^{\frac{n-1}{2}} \psi(2^{n-1} t - k) dt. \tag{2.74}$$

Ahora, de la ecuación básica de recursión (2.38) es posible obtener una representación tanto para $\phi(2^{n-1}t - k)$ como para $\psi(2^{n-1}t - k)$ así:

$$\phi(2^{n-1}t - k) = \sum_p h(p) \sqrt{2} \phi(2^n t - 2k - p) \tag{2.75}$$

$$\psi(2^{n-1}t - k) = \sum_p h_1(p) \sqrt{2} \phi(2^n t - 2k - p) \tag{2.76}$$

Reemplazando las dos últimas expresiones en las integrales de (2.73) y (2.74) y haciendo un cambio de variable $m = 2k + p$ se obtiene:

$$c_{n-1,k} = < f(t), \phi_{n-1,k}(t) > = \int_{-\infty}^{\infty} f(t) \sum_m h(m - 2k) 2^{\frac{n}{2}} \phi(2^n t - m) dt \tag{2.77}$$

$$d_{n-1,k} = < f(t), \psi_{n-1,k}(t) > = \int_{-\infty}^{\infty} f(t) \sum_m h_1(m - 2k) 2^{\frac{n}{2}} \phi(2^n t - m) dt \tag{2.78}$$

o lo que es lo mismo:

$$c_{n-1,k} = < f(t), \phi_{n-1,k}(t) > = \sum_m h(m-2k) \int_{-\infty}^{\infty} f(t) 2^{\frac{n}{2}} \phi(2^n t - m) dt \tag{2.79}$$

$$d_{n-1,k} = < f(t), \psi_{n-1,k}(t) > = \sum_m h_1(m-2k) \int_{-\infty}^{\infty} f(t) 2^{\frac{n}{2}} \phi(2^n t - m) dt \,. \tag{2.80}$$

Fácilmente se observa que en (2.79) y (2.80) las integrales son idénticas y corresponden al coeficiente $c_{n,m}$, es decir, se obtuvo una representación de los coeficientes escala y *wavelets* en un nivel de resolución más bajo en términos de los coeficientes escala en un nivel de resolución más alto. Una descripción matemática de lo anteriormente expuesto se muestra en las relaciones (2.81) y (2.82).

$$c_{n-1,k} = \sum_m h(m-2k) c_{n,m} \tag{2.81}$$

$$d_{n-1,k} = \sum_m h_1(m-2k) c_{n,m} \,. \tag{2.82}$$

Se observa entonces que la secuencia de entrada dada por $c_{n,k}$ se convoluciona con h y h_1 para obtener por una lado una representación más "suave" de la señal original representado por los coeficientes escala $c_{n-1,k}$, y por otro lado el detalle de la señal representado por los *coeficientes wavelet* $d_{n-1,k}$. Con ello (2.72) toma la forma:

$$f(t) = \sum_k c_{n-1,k} 2^{\frac{n-1}{2}} \phi(2^{n-1} t - k) + \sum_k d_{n-1,k} 2^{\frac{n-1}{2}} \psi(2^{n-1} t - k) \,. \tag{2.83}$$

Como se ha mencionado en varias ocasiones, los *coeficientes escalares* representan la forma general de la señal original y los *coeficientes wavelets* el detalle de la misma, pues bien, este hecho se debe a que los coeficientes h y h_1 actúan como filtros digitales. Específicamente h actúa como un filtro pasa-bajo y h_1 como un filtro pasa-alto, sin embargo, al aplicar estas operaciones sobre una señal digital real, resulta a la salida una señal con el doble de datos de entrada. Lo anterior se soluciona aplicando una operación denominada submuestreo justamente después de efectuar la convolución discreta sobre el conjunto de datos de entrada, esta operación realiza un diezmado de la señal original, es

decir, si se tiene una señal x_n el submuestreo produce una salida $y_n = x_{2n}$, o sea descarta todos los valores de índice impar. En la figura 9 se representa la situación expuesta, mostrando una *descomposición wavelet*, en donde el 2 con la flecha hacia abajo representa la operación del submuestreo.

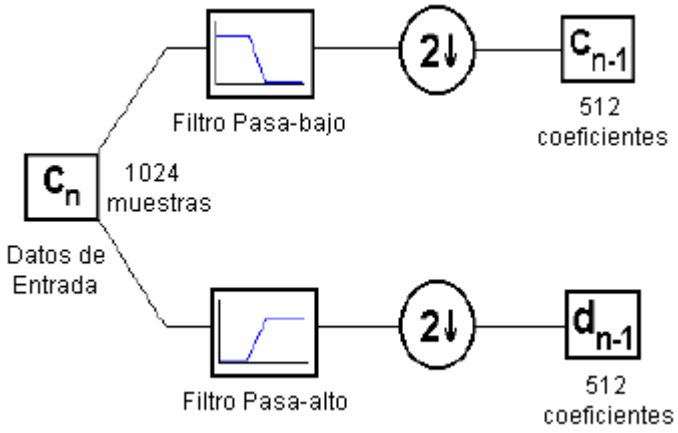

Figura 9. ***Descomposición wavelet***

2.4.4.2 Reconstrucción de señales unidimensionales (Síntesis)

Hasta el momento se ha estudiado la forma cómo opera la DWT para analizar o descomponer una señal, ahora bien, interesa en este momento saber el proceso inverso, es decir, cómo recuperar la señal original sin pérdida de información a partir de las componentes obtenidas durante el análisis, a este proceso de reconstrucción se le denomina *síntesis* y corresponde a la inversa de la *transformada discreta wavelet* (IDWT). El proceso de síntesis busca representar los coeficientes escala en un nivel de resolución más alto mediante una combinación de los coeficientes escala y *wavelets* en un nivel de resolución más bajo.

Con el propósito de estudiar más a fondo el proceso de síntesis, observe que si se utiliza la ecuación de recursión (2.38) para reemplazar $\phi(2^{n-1}t - m)$ y $\psi(2^{n-1}t - m)$ en (2.83) se obtiene una nueva expresión para *f* así:

$$f(t)=\sum_m c_{n-1,m}\sum_p h(p)2^{\frac{n}{2}}\phi(2^n t-2m-p)+\sum_m d_{n-1,m}\sum_p h_1(p)2^{\frac{n}{2}}\phi(2^n t-2m-p) \qquad (2.84)$$

luego, si se multiplica ambos lados de la anterior ecuación por $\phi(2^n t-k)$ e integrando con respecto al tiempo se llega a:

$$\begin{aligned}\int_{-\infty}^{\infty} f(t)\phi(2^n t-k)dt &= \sum_m c_{n-1,m}\sum_p h(p)2^{\frac{n}{2}}\int_{-\infty}^{\infty}\phi(2^n t-2m-p)\phi(2^n t-k)dt \\ &+\sum_m d_{n-1,m}\sum_p h_1(p)2^{\frac{n}{2}}\int_{-\infty}^{\infty}\phi(2^n t-2m-p)\phi(2^n t-k)dt.\end{aligned} \qquad (2.85)$$

Ahora, como $c_{n,k}=<f(t),\phi_{n,k}(t)>=\int_{-\infty}^{\infty} f(t)2^{\frac{n}{2}}\phi(2^n t-k)dt$ y el conjunto $\{\phi_{n,k};k\in\square\}$ es ortonormal, entonces de acuerdo con (2.85) se tiene que:

$$c_{n,k}=\sum_m c_{n-1,m}\sum_p h(p)\delta(k-(2m-p))+\sum_m d_{n-1,m}\sum_p h_1(p)\delta(k-(2m+p)), \qquad (2.86)$$

si se realiza la sustitución $q=2m-p$ se llega a:

$$c_{n,k}=\sum_m c_{n-1,m}\sum_p h(p)\delta(k-q)+\sum_m d_{n-1,m}\sum_p h_1(p)\delta(k-q) \qquad (2.87)$$

pero además se tiene que:

$$\delta(k-q)=\begin{cases}1 & si\ k=q\\ 0 & si\ k\neq q\end{cases}$$

entonces finalmente se obtiene la expresión para $c_{n,k}$ dada en (2.88).

$$c_{n,k}=\sum_m c_{n-1,m}h(2m-k)+\sum_m d_{n-1,m}h_1(2m-k) \qquad (2.88)$$

Observe que en el proceso de análisis se hace un filtrado y un submuestreo, de igual manera en el proceso de síntesis se realiza un supermuestreo y posteriormente un filtrado. El supermuestreo es una operación que inserta ceros entre cada par de muestras con el fin de aumentar al doble la longitud de las componentes de entrada (coeficientes escala y

wavelet) de tal manera que la señal obtenida después del filtrado tenga la misma longitud que la señal original. En la figura 10, se representa lo expuesto anteriormente, mostrando la *reconstrucción wavelet* donde el 2 con la flecha hacia arriba representa la operación del supermuestreo.

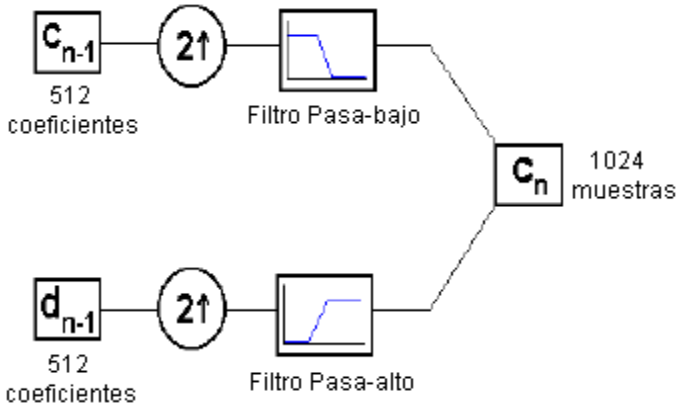

Figura 10. Reconstrucción *wavelet*

Los procesos de análisis y síntesis explicados anteriormente son iterativos y en la práctica el nivel de resolución de la señal original es el que determina el límite de esa iteración. Estos dos procesos constituyen lo que se denomina un *sistema de banco de filtros de 2 canales.*

Para entender mejor el proceso iterativo del cual se hizo mención, sea una señal con una longitud $N = 2^n$, en el análisis se divide la señal original en una aproximación y un detalle correspondientes al primer nivel de descomposición, luego la aproximación de longitud igual a 2^{n-1} se divide nuevamente obteniendo una nueva aproximación y detalle correspondientes a un segundo nivel de descomposición, se continúa este procedimiento hasta que la aproximación y el detalle sean representados por un sólo coeficiente, es decir, hasta que tenga una longitud de $1 = 2^0$.

Con lo anterior fácilmente se deduce que el número de iteraciones posibles de realizar es de $n = \log_2 N$. Finalmente se obtiene un vector de longitud N que contiene en su primera componente un término que representa la forma general de la señal (coeficiente escala) y todos los demás componentes representan la información sobre el detalle obtenido en los

diferentes niveles de descomposición (*coeficientes wavelets*), a este vector se le denomina *vector DWT*. Una ilustración gráfica de lo anteriormente expuesto se representa en la figura 11.

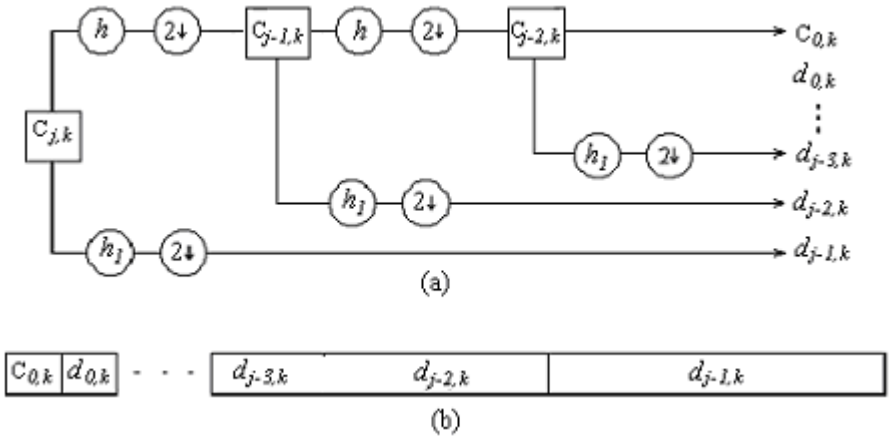

Figura 11. (a) Proceso de análisis, (b) vector - DWT

Por otro lado, el proceso de síntesis realiza el procedimiento inverso al previamente explicado, es decir, toma la aproximación y el detalle, aumenta la longitud al doble mediante el supermuestreo y realiza la convolución discreta con los respectivos filtros, obteniéndose como resultado una mejor aproximación a la señal correspondiente al primer nivel de reconstrucción. Lógicamente el número de veces que se realiza este proceso hasta llegar nuevamente a la señal original depende del grado de descomposición al que se llegó en el análisis. En la figura 12 se muestra este proceso de síntesis.

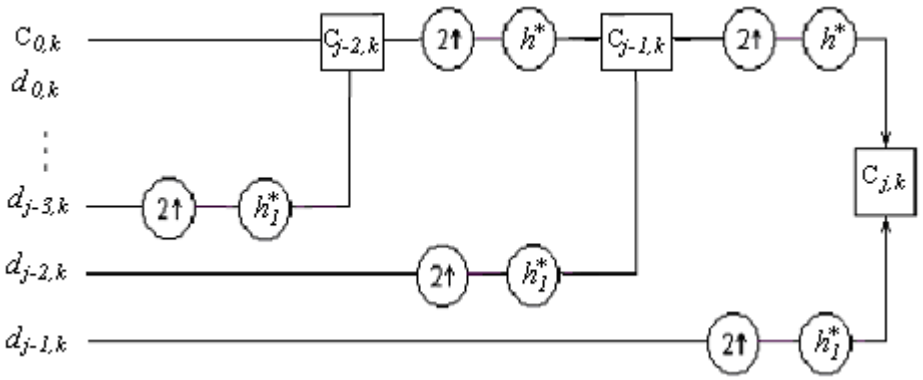

Figura 12. Proceso de síntesis

3 COMPRESIÓN Y RECONSTRUCCIÓN DE IMÁGENES

En el capítulo anterior se expuso el referente teórico más importante requerido para el entendimiento de la *teoría de wavelets* que a su vez se convierte en la base fundamental para el desarrollo de una de las aplicaciones de mayor renombre de dicha teoría: la comprensión de imágenes. En este capítulo se presenta lo concerniente a la compresión y reconstrucción de imágenes con el propósito de abordar en los capítulos posteriores el desarrollo de una aplicación de simulación para comprimir imágenes fijas.

La importancia de esta aplicación de la *teoría de wavelets* radica en el hecho que dentro de la sociedad actual, caracterizada por las redes de comunicaciones, servicios de multimedia, computadores, fax, televisión, sistemas de teleconferencia, entre muchos otros, los tipos de información que más capacidad exigen son las imágenes y el vídeo.

Como contribución a los esfuerzos de los últimos años orientados al mejoramiento en la eficiencia de la transmisión de datos a través de las redes y la reducción del coste de almacenamiento, los avances en el área de la compresión de datos ha sido de gran importancia, y entre las distintas técnicas disponibles en la actualidad, la de compresión de imágenes con *wavelets* juega un papel fundamental [10].

3.1 CONCEPTOS SOBRE IMÁGENES DIGITALES

Antes de abordar lo que es la compresión de imágenes como tal, se requiere para su claro entendimiento hacer una revisión de conceptos básicos sobre las características de las imágenes digitales.

3.1.1 Representación de una imagen

Una imagen puede ser representada mediante una función bidimensional de intensidad de luz $f(x,y)$ donde el valor de la función en un punto determinado es proporcional al brillo o nivel de gris de la imagen en ese punto [11].

Por otro lado, una imagen digital dada por $f(x,y)$ es la que se discretiza tanto en las coordenadas como en los niveles de gris, la cual es representada por una matriz $M \times N$ constituida por los valores enteros no negativos $f(x,y)$, donde $1 \leq x \leq M$ y $1 \leq y \leq N$ son las coordenadas de la imagen donde la ubicación de cada componente (llamado píxel) representa un punto de la imagen, y el valor de la misma se conoce como valor funcional o nivel de gris en ese punto.

Ahora bien, en el procesamiento de imágenes, dos aspectos fundamentales a tener en cuenta son el *muestreo* y la *cuantificación del nivel de gris* de una imagen. El primero se refiere a la discretización de las coordenadas espaciales y el segundo a la discretización de la amplitud. Partiendo de este concepto, es posible representar en forma aproximada una imagen continua por una serie de muestras igualmente espaciadas, y esto a la vez se puede representar por medio de una matriz [12]. Así, la función $f(x,y)$ está dada por dicha matriz de la siguiente manera:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ \vdots & & & & \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1,N-1) \end{bmatrix} \tag{3.1}$$

3.1.2 Almacenamiento de una imagen digital

El almacenamiento de información de cualquier tipo, está determinado por el número de bits de memoria necesarios para almacenar tal información. En el caso de una imagen digital, dicha cantidad depende lógicamente de los valores *M* y *N*. Normalmente estos valores son potencias de *2* ($M = 2^n$ y $N = 2^k$ con *n* y *k* enteros); ahora bien, el número de niveles de gris (o niveles de brillo) de una imagen (notado con la letra *G*) también es una potencia de *2* ($G = 2^m$ donde *m* es entero y se conoce como la profundidad de los niveles de gris). De esta manera, la cantidad de bits (*b*) necesarios para almacenar una imagen digital se determina de acuerdo a la siguiente ecuación:

$$b = M \times N \times m \tag{3.2}$$

Cuando la matriz que representa la imagen es cuadrada, es decir, cuando $M = N$, se tiene

$$b = N^2 \times m \tag{3.3}$$

Obviamente si el número de píxeles y el número de niveles de grises aumenta, también aumenta el espacio en memoria requerido para almacenar la imagen respectiva. Por ejemplo, una imagen de 512 x 512 píxeles a 256 niveles de grises necesita 2.097.152 bits de memoria para ser almacenada [11].

3.1.3 Resolución de una imagen digital

De lo explicado anteriormente, se deduce que el muestreo y la cuantificación del nivel de gris de una imagen digital dependen del número de píxeles y de niveles de gris de la misma, es decir, al aumentar estos parámetros no solo aumenta el tamaño de memoria requerido para su almacenamiento como lo muestra la ecuación (3.3), sino también, se hacen mayores los requerimientos de procesamiento. No obstante, entre mayores sean el muestreo y la cuantificación del nivel de gris, más se aproxima una imagen digitalizada a la original. La resolución es una importante propiedad de una imagen y se encuentra estrechamente relacionada con estos parámetros.

La apariencia de una imagen digitalizada depende entonces de la resolución, definida como el número de píxeles por unidad lineal, medida en *dpi* (*puntos por pulgada*). Por ejemplo, un fax maneja una resolución de 200 *dpi* en dirección horizontal y 100 *dpi* en dirección vertical [13].

En la digitalización de imágenes, la decisión de cuántos *dpi* utilizar regularmente se rige por consideraciones prácticas. Mientras más grande es el número de *dpi*, mayor información tendrá un archivo y el detalle de la imagen será más fino, es en este punto donde juega un papel importante la calidad de la imagen dependiendo de las necesidades de la aplicación.

3.1.4 Entropía

Sea una fuente de información sin memoria (los símbolos emitidos por la fuente son estadísticamente independientes) que usa un alfabeto $\{a_k\}$, $k = 0,1,...,k-1$, donde a_k son los símbolos del alfabeto, el teorema de Shannon establece que la información *I* que aporta una ocurrencia de un símbolo a_k en un mensaje está definido por [14]:

$$I(a_k) = -\log\left[P(a_k)\right] \tag{3.4}$$

donde $P(a_k)$ es la probabilidad de ocurrencia del símbolo a_k.

Ahora bien, la *entropía* de una fuente de información representa el contenido de información media de los mensajes generados por la fuente y está dada mediante la expresión (3.5).

$$H = E\left\{I(a_k)\right\} = -\sum P(a_k)\log\left[P(a_k)\right] \tag{3.5}$$

La unidad de información está determinada por la base del logaritmo; si la base es 2, entonces la entropía se mide en *bits*, si es 10, la entropía se mide en *hartleys*.

3.2 COMPRESIÓN DE IMÁGENES

En ciertos casos la representación digital de una imagen involucra una gran cantidad de datos que puede llegar a ser tan extensa que dificulta el procesamiento, almacenamiento y hasta su transmisión. Es por esta razón que se hace necesario reducir la cantidad de datos mediante la compresión.

El proceso de codificación de imágenes en un principio se realizaba de una manera muy sencilla en el que para cada píxel se leía/escribía el índice correspondiente a su color dentro de la gama de colores el cual se codificaba. Estos ficheros de imágenes eran fáciles de codificar y de decodificar y por tanto eran procesos muy rápidos pero poco eficientes [15].

A medida que las imágenes aumentaron su tamaño, resolución y su profundidad de color, los archivos que las almacenaban crecieron de manera proporcional, fue entonces cuando se dio mayor relevancia al tratamiento digital de imágenes y se encontró que la mayoría de ellas poseían mucha información repetitiva (por ejemplo, grandes áreas donde el color de los píxeles era el mismo que el de sus vecinos), un alto grado de redundancia espacial, y una alta correlación entre los valores de los píxeles, lo cual se convirtió en el fundamento del tema de la compresión, tan importante en la actualidad.

En este sentido y en pocas palabras puede definirse a la compresión como la "*supresión de información redundante"* que trata de aprovechar dicha redundancia para reducir el número de bits necesarios para representar la imagen, consiguiendo con ello ahorrar recursos tanto de almacenamiento como de transmisión.

En la compresión digital de imágenes, se identifican y aprovechan 3 tipos básicos de redundancias: redundancia de codificación, redundancia entre píxeles y la redundancia psicovisual [16, 17].

3.2.1 Redundancia de codificación

Los niveles de gris de una imagen digital se encuentran en el rango que va desde *0* a 2^b, donde *b* representa los bits utilizados para la escala de gris respectiva. Entonces, si r_k representa el k-ésimo nivel de gris de la imagen, la probabilidad de que aparezca ese nivel de gris se denota por $p_r(r_k)$, la cual se calcula mediante la siguiente expresión:

$$p_r = \frac{n_k}{n}, \quad k = 0,1,...,L-1 \tag{3.6}$$

donde n_k es el número de píxeles de la imagen con ese nivel de gris, *n* el número total de píxeles de la imagen y *L* es el número de nivel de gris ($L = 2^b$). De otro lado, si el número de bits empleados para representar cada valor de r_k es $l(r_k)$, el promedio de bits necesarios para representar cada píxel, o lo que es lo mismo, la longitud media de las palabras de código asignadas a los valores de los diferentes niveles de gris está dado por:

$$L_{med} = \sum_{k=0}^{L-1} l(r_k)\, p_r(r_k) \tag{3.7}$$

Es decir, dicha longitud se calcula sumando el producto del número de bits empleados para representar cada nivel de gris y la probabilidad de que aparezca ese nivel. Con lo anterior, si se tiene una imagen $M \times N$, el número total de bits necesarios para codificarla es $M \times N \times L_{med}$, como se mencionó en la sección (3.1.2).

Ahora bien, considere dos conjuntos de datos, el primero con n_1 unidades de información y el segundo con n_2, y suponga que los dos conjuntos representan la misma información. Entonces, la relación de compresión C_R está dada por la expresión (3.8).

$$C_R = \frac{n_1}{n_2} \tag{3.8}$$

y la redundancia relativa de los datos R_D del primer conjunto de datos (caracterizado por n_1) se define como:

$$R_D = 1 - \frac{1}{C_R} \tag{3.9}$$

3.2.2 Redundancia entre píxeles

Este tipo de redundancia se refiere a la posibilidad de predecir razonablemente el valor de un determinado píxel a partir del valor de sus vecinos, ya que la información que aporta individualmente un píxel podría ser relativamente pequeña, dependiendo claro está, de la imagen y de la posición de dicho píxel dentro de la misma. La mayor parte de la contribución visual de un único píxel a una imagen es redundante y puede ser inferido de acuerdo con los valores de los píxeles vecinos. Esta dependencia entre píxeles es lo que se denomina redundancia entre píxeles, conocida también como redundancia espacial, redundancia geométrica o redundancia interna.

3.2.3 Redundancia psicovisual

En el proceso visual normal humano hay cierta información que tiene menor importancia relativa que otra ya que el ojo humano no responde con la misma sensibilidad a toda la información visual. Se dice que esa información es psicovisualmente redundante, sin la cual la calidad de la percepción de la imagen no se alteraría en forma significativa.

En ese proceso visual, un observador busca ciertas características significativas y diferenciadoras tales como la diversidad de texturas o bordes, para que luego mentalmente se realice el proceso de interpretación de la imagen; esto es posible gracias a la sensibilidad inherente del ojo a este tipo de características.

De acuerdo a lo anterior, la redundancia psicovisual está asociada a esa información visual real o cuantificable, la cual es posible eliminar debido a que la propia información no es esencial para dicho procesamiento visual; ahora, al eliminar los datos psicovisualmente redundantes se presenta una pérdida de información cuantitativa por lo cual a veces se le

denomina *cuantificación*, es decir que a un gran rango de valores de entrada le corresponde un limitado rango de valores de salida implicando entonces una compresión con pérdida de datos.

3.2.4 Medidas de la compresión

Algunos de los conceptos más importantes e inherentes a un proceso de compresión se explican a continuación [18].

- *Longitud Media Por Símbolo* (*LMPS*): Es la relación entre el peso del archivo comprimido (*C*) en *Bytes* y el del archivo sin comprimir (*O*) expresado en *Bytes*. Esta relación se muestra en la ecuación (3.10).

$$LMPS = \frac{C}{O}.8\ bps \tag{3.10}$$

- *Factor de Compresión* (*FC*): Es la relación entre el peso del archivo original y el del comprimido como se muestra en la ecuación (3.11).

$$FC = \frac{O}{C} \tag{3.11}$$

- *Compresión Relativa* (*CR*): Se define como la diferencia entre el peso del archivo original y el peso del archivo comprimido (el peso del archivo comprimido que disminuye respecto del original). Se mide a través de la siguiente relación:

$$CR = \frac{O - C}{O} \tag{3.12}$$

3.3 TÉCNICAS DE COMPRESIÓN DE IMÁGENES

Existen diversas formas de clasificar las técnicas de compresión[1] de imágenes digitales según la característica específica que se quiera diferenciar. Una de esas formas y posiblemente la más general establece dos grandes grupos:

- *Compresión sin pérdidas* o *lossless* (conocidas también como *reversibles* o '*noiseless*')
- *Compresión con pérdidas o lossy* (conocidas también como *Irreversibles* o '*noisy*').

La compresión *sin pérdidas* implica que al reconstruir una imagen luego del proceso de compresión, ésta coincide exactamente con la imagen original hasta en el más pequeño detalle. En otras palabras, mediante este tipo de técnicas no se pierde información.

Por el contrario, la compresión *con pérdidas* se fundamenta en suprimir cierta información de la imagen, específicamente detalles no significativos, conservando un nivel arbitrario de la calidad de la misma (de acuerdo a criterios de fidelidad de imagen como los considerados en la sección 3.7) para hacerla "más liviana", de este modo, la imagen reconstruida no es idéntica a la imagen original. Es así como mediante este tipo de técnicas se obtienen factores de compresión del orden de 10:1, 50:1 o mayores, factores muy superiores a los obtenidos a través de métodos *lossless*, los cuales están alrededor de 2:1 o 5:1. Lógicamente un mayor factor de compresión implica mayor degradación de la imagen, es por ello que en la compresión *lossy* debe establecerse un compromiso o "*tradeoff* " entre la calidad de la imagen y dicho factor, dependiendo de la aplicación específica para la cual se requiere el proceso de compresión.

Algunos de los métodos clasificados dentro de los dos grupos de técnicas de compresión mencionados se presentan en las siguientes secciones.

[1] Tanto la compresión como la reconstrucción son operaciones llamadas de *codificación de imágenes*, debido a que hacen uso de métodos de codificación de datos, para representar una imagen de una forma más concisa.

Las técnicas de compresión de imágenes ya sea con o sin pérdidas pueden modelarse en forma general como se representa en la figura 13.

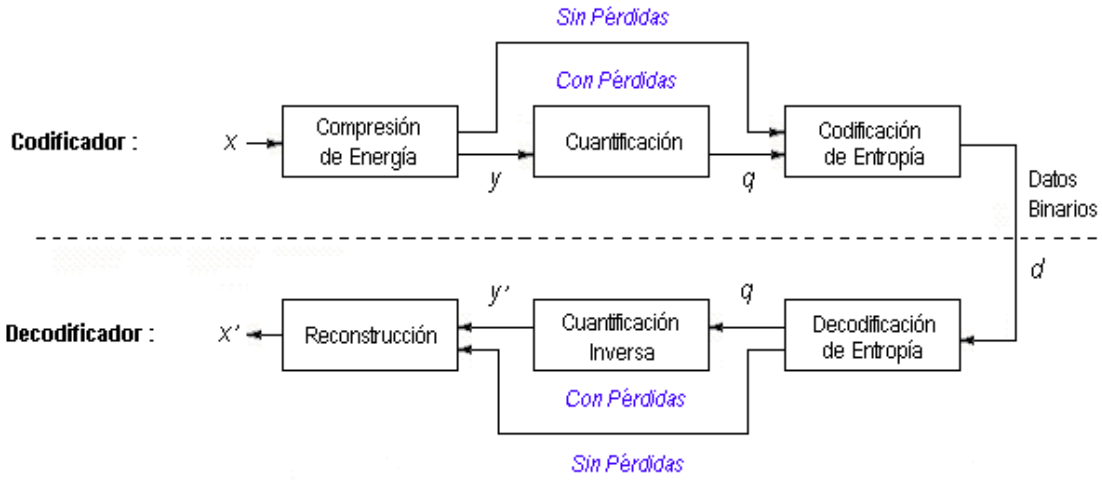

Figura 13. Esquema general de un sistema de compresión de imágenes

En el codificador se puede diferenciar tres etapas básicas [19]:

- *Compresión de energía*: consiste usualmente en un proceso de transformación o de filtrado que busca concentrar tanto como sea posible una alta proporción de energía de la imagen *x* a comprimir en unas pocas muestras (coeficientes estadísticamente independientes) mientras que se preserva el total de la energía. Además, esta etapa supone una decorrelación de la señal, es decir, busca reducir la redundancia entre píxeles.
- *Cuantificación*: Es la etapa que controla el grado de distorsión así como la tasa de bits deseada. Consiste en reducir la precisión de los coeficientes *y* transformarlos en una matriz *q* sin sobrepasar los límites de calidad estimados aprovechando la redundancia psicovisual de la imagen. Esta fase implica siempre algún tipo de pérdida, por ende, las técnicas *lossless* carecen de ella.
- *Codificación de entropía*. Aumenta la compresión sin aumentar las pérdidas codificando los coeficientes transformados de forma más compacta en una cadena de bits *d*, en otras palabras, busca reducir la redundancia de codificación de la imagen. Tras esta codificación se tiene una cadena de bits sin separación distinguible en lugar de una serie de coeficientes o símbolos.

Entre tanto, en el decodificador se realiza en cada una de las etapas el proceso inverso al realizado en el codificador, así por ejemplo, en la etapa de cuantificación inversa se reconstruye y' como la mejor estimación de y a partir de q.

Las etapas de compresión de energía, reconstrucción, y codificación/decodificación de entropía, son normalmente sin pérdidas. Únicamente la etapa de cuantificación introduce pérdida y distorsión, donde y' es una versión distorsionada de y, por lo tanto x' es un versión distorsionada de x, mientras que en la ausencia de cuantificación, se tiene que si $y' = y$, entonces $x' = x$. Ver figura 13.

3.3.1 Métodos de compresión sin pérdidas (*Lossless*)

El estado del arte referente a este tipo de compresión presenta una gran variedad de métodos, muchos de los cuales provienen directamente del mundo de compresión de datos y se han adaptado para el uso con datos digitales de imágenes. Entre los más difundidos se encuentran [20]:

- Codificación de longitud fija
- Codificación por entropía o de longitud variable
- Codificación de planos de bits
- Codificación por longitud de series (RLE)
- Codificación predictiva sin pérdidas
- Codificación del contorno
- Codificación por bloques sin pérdidas

A continuación y con el propósito de establecer el referente teórico necesario para el desarrollo de este proyecto, se explica solo la codificación de longitud variable, con la que entre otras cosas, se ha logrado mejores resultados.

3.3.1.1 Codificación de longitud variable

Este método se conoce también como *codificación de entropía* puesto que trata de aproximar *la longitud media de las palabras de código* a la entropía del sistema. Consiste en disminuir la redundancia de codificación normalmente presente en cualquier codificación binaria natural de los niveles de brillo (es decir, los niveles de gris) de una imagen. Para ello, la técnica utilizada es asignar un número pequeño de bits a aquellos niveles más probables y mayor número de bits a los menos probables [21].

Existen varios métodos de codificación de longitud variable, pero los más usados son la codificación Huffman y la codificación aritmética, siendo esta última la que mejores factores de compresión proporciona (normalmente alcanza entre un 5% y un 10% de mejor compresión respecto a la Huffman), aunque es más compleja que la primera. Para el desarrollo de este proyecto se implementa la codificación aritmética, sin embargo, a continuación se describe el funcionamiento de la codificación Huffman puesto que es una de las más difundidas y utilizadas en la compresión de datos.

- **Codificación Huffman**

Es una de las técnicas más populares para la eliminación de la redundancia de la codificación en la cual se codifica individualmente los símbolos de una fuente de información. La codificación Huffman convierte los valores de brillo de los píxeles en nuevos códigos de longitud variable, basados en su frecuencia de aparición en la imagen. Así, a los brillos que aparecen con mayor frecuencia se les asigna códigos más cortos, y a los menos frecuentes, códigos más largos.

El primer paso de esta codificación es crear una serie de reducciones de la fuente para lo cual se ordena las probabilidades respectivas de los símbolos, luego se combinan los dos símbolos de menor probabilidad creando uno nuevo que los sustituye en la siguiente reducción de la fuente, donde su probabilidad es la suma de las probabilidades de los

símbolos operados y se reordena de nuevo descendentemente, este proceso se repite hasta que se consigue una fuente reducida con dos símbolos, es decir hasta llegar a la probabilidad unidad.

La segunda etapa del procedimiento de Huffman consiste en recorrer la cadena hacia atrás asignando los valores 1 y 0 (1 a la menor probabilidad y 0 a la mayor) a cada rama de la unión, concatenándolo con los valores asignados anteriormente, de tal forma que el código para cada símbolo inicial se forma recorriendo su trayectoria de derecha a izquierda [16, 18, 21, 22]. La figura 14 ilustra el proceso completo.

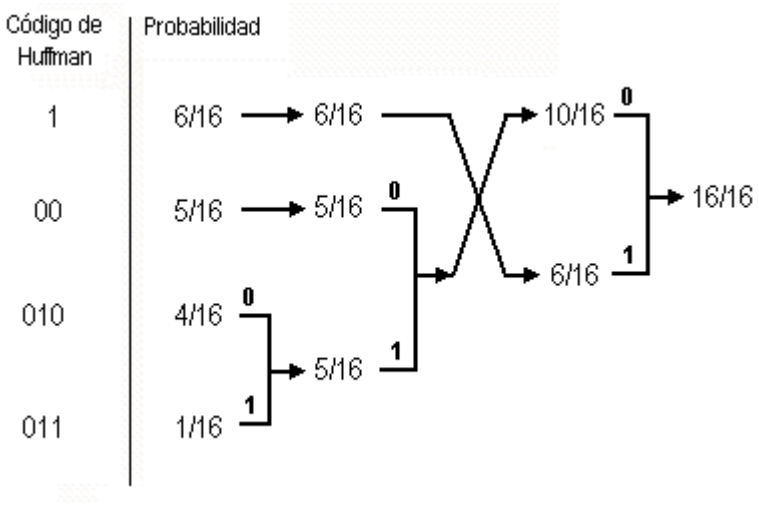

Figura 14. Ejemplo de codificación Huffman

Es importante destacar que tal como se forman las palabras de código, no es posible que coincida un segmento inicial de una de ellas con otra completa. Además, si el cero es utilizado como símbolo aislado, entonces no se emplea como principio de ningún otro símbolo y así con cualquier combinación, esto evita cualquier confusión en el proceso de decodificación así no se indique explícitamente el límite entre símbolos, siempre y cuando se conozcan las palabras códigos fijadas. Adicionalmente, el código Huffman más largo nunca puede ser mayor que el número de niveles de brillo diferentes en la imagen menos 1, y aunque una imagen codificada con el método de Huffman puede tener un poco de brillo con códigos muy largos, sus frecuencias de ocurrencia siempre son estadísticamente bajas.

Generalmente, la compresión de imágenes mediante Codificación Huffman proporciona factores de compresión entre 1.5:1 y 2:1.

- **Codificación Aritmética**

Al contrario de lo que sucede con la codificación Huffman, la codificación aritmética no genera códigos bloque. En esta técnica no existe una correspondencia biunívoca entre los símbolos de la fuente y las palabras código. Consiste en que se asigna una sola palabra código aritmética a una secuencia completa de símbolos fuente, la cual define un intervalo de números reales comprendido entre 0 y 1 [11, 16, 21].

Una característica importante de esta codificación es que a medida que aumenta el número de símbolos del mensaje, el intervalo que se utiliza para representarlo se va haciendo menor y se va incrementando el número de unidades de información necesarias para representar dicho intervalo. Cada símbolo del mensaje reduce el tamaño del intervalo según su probabilidad de aparición.

La codificación aritmética, a diferencia de la codificación Huffman, no requiere que cada símbolo de la fuente se traduzca en un número entero de símbolos de código, es decir, aquí los símbolos no se codifican uno a uno.

En la figura 15 se muestra el proceso básico de este tipo de codificación en el que se codifica una secuencia de cinco símbolos $a_1\ a_2\ a_3\ a_3\ a_4$, generados por una fuente de cuatro símbolos con probabilidades 0.2, 0.2, 0.4 y 0.2 respectivamente. Al comienzo del proceso se supone que el mensaje ocupa todo el intervalo $[0,1)$. Luego este intervalo se subdivide en cuatro partes en función de las probabilidades de cada símbolo, por ejemplo se asocia el intervalo $[0,0.2)$ al símbolo a_1.

Ahora bien, debido a que este símbolo es el primero en codificar, el intervalo del mensaje se reduce precisamente al mencionado, es por esta razón que el intervalo $[0,0.2)$ abarca toda la altura de la figura y se marcan los extremos con los valores del mismo, luego de esto, se divide nuevamente este rango reducido en función de nuevo de las probabilidades de los símbolos del mensaje original.

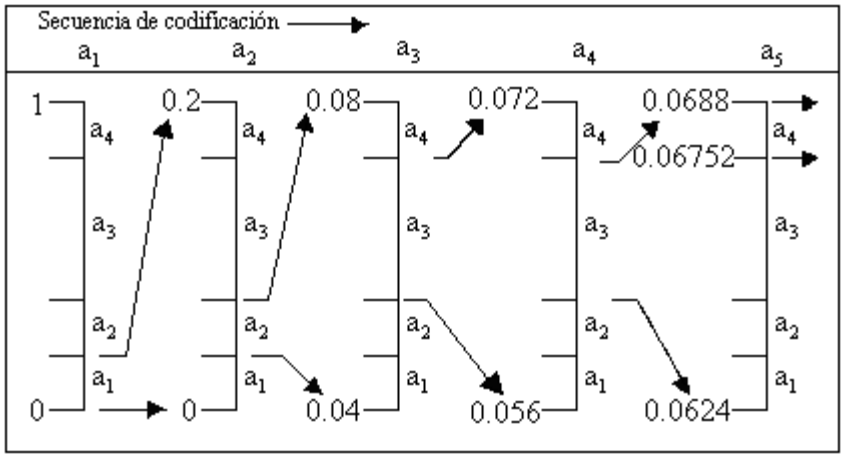

Figura 15. Ejemplo de codificación aritmética

Así, el procedimiento continúa con cada símbolo en su orden de codificación reduciendo el intervalo cada vez más. Finalmente se toma como codificación del mensaje el límite inferior del último intervalo reducido y que corresponde obviamente al último símbolo en codificar. Ver figura 15.

3.3.2 Métodos de compresión con pérdidas (Lossy)

Este tipo de compresión se caracteriza porque existen diferencias entre la imagen original y la imagen obtenida luego del proceso de reconstrucción, debido a la realización de modificaciones irreversibles en la imagen, las cuales tienen lugar en la etapa de cuantificación. Dentro de este grupo es importante diferenciar las técnicas denominadas

visualmente sin pérdidas[2] donde a pesar de suprimir información de la imagen, la diferencia no es detectada por el ojo humano.

Todos los esquemas de compresión de imágenes con pérdidas involucran la eliminación de datos de la imagen, para lo cual, la imagen primero se transforma a otra, y entonces se suprimen partes de ella. Son precisamente los métodos empleados para transformar y suprimir datos de la imagen los que caracterizan y diferencian a cada uno de los esquemas respecto a los demás dentro de esta categoría.

Se han desarrollado muchos esquemas de compresión de imágenes con pérdidas, generalmente, cada uno cumple con los requisitos de calidad de una aplicación específica. Entre los más difundidos y aplicados se encuentran [20]:

- Codificación por truncamiento
- Codificación predictiva diferencial con pérdidas
- Codificación por Transformada
- Codificación por bloques con pérdidas

Considerando la utilidad dentro del proyecto, en la siguiente sección se presenta el esquema de codificación por transformada, dando especial importancia a la *transformada wavelet*.

3.3.2.1 Codificación por transformada

En la codificación por transformada se realiza sobre la imagen algún tipo de transformación para tratar de eliminar la redundancia entre píxeles antes de llevar a cabo el proceso de cuantificación. De este modo, lo que se codifica no es la imagen como tal, sino sus coeficientes en el espacio transformado. Este tipo de codificación puede modelarse de acuerdo a la figura 13 que se mostró en la sección 3.3 donde el módulo de *compresión de energía*, corresponde al proceso de aplicación de una transformada matemática que permite

[2] De acuerdo a estas técnicas, se puede obtener factores de compresión del orden de 10:1

llevar la imagen del dominio espacial a otro dominio, generalmente el de la frecuencia[3]. En este dominio la mayoría de la energía se concentra en un número pequeño de componentes, los cuales se debe cuantificar con mayor resolución.

En el dominio de la frecuencia, los componentes fundamentales representados por los brillos de los píxeles, generalmente tienden a estar agrupados en regiones o zonas de baja de frecuencia. Como resultado, suele haber grandes áreas de la imagen transformada, donde los componentes tienen un valor muy pequeño, consecuencia del proceso de transformación al dominio de la frecuencia que elimina mucha redundancia de la imagen. La compresión se logra entonces eliminando aquellos componentes de pequeño valor, mientras que los componentes de frecuencia restantes se convierten en los códigos que son almacenados para formar la imagen comprimida [16].

La operación de reconstrucción consiste simplemente en la transformación inversa de los componentes de frecuencia que quedaron almacenados. La calidad de la imagen reconstruida depende de qué componentes y cuántos se eliminaron de la imagen transformada, así mismo, se ve afectada por la reducción de resolución de los componentes no eliminados. Generalmente se obtienen relaciones de compresión elevadas, tal como se mencionó en la sección 3.3, manteniendo buena calidad.

Aunque hay diversas clases de transformadas que pueden utilizarse para un sistema de este tipo, la transformada más utilizada en la codificación de imágenes es la DCT[4], dada la existencia de algoritmos de cálculo rápidos y su alta eficiencia, sin embargo, recientemente la *transformada wavelet*[5], cuyas bondades se ilustran en este trabajo, aparece como una mejor opción.

[3] Al dominio tiempo-frecuencia o tiempo-escala en el caso de la *trasformada wavelet*.

[4] Empleada en el estándar de compresión de imágenes JPEG.

[5] Empleada en el estándar de compresión de imágenes JPEG 2000.

- **Compresión por DCT**

La introducción en 1974 de la transformada discreta coseno (DCT: d*iscrete cosine transform*) contribuyó de manera importante al desarrollo de la teoría de compresión de imágenes. La codificación mediante la DCT, se basa en el esquema de la figura 16.

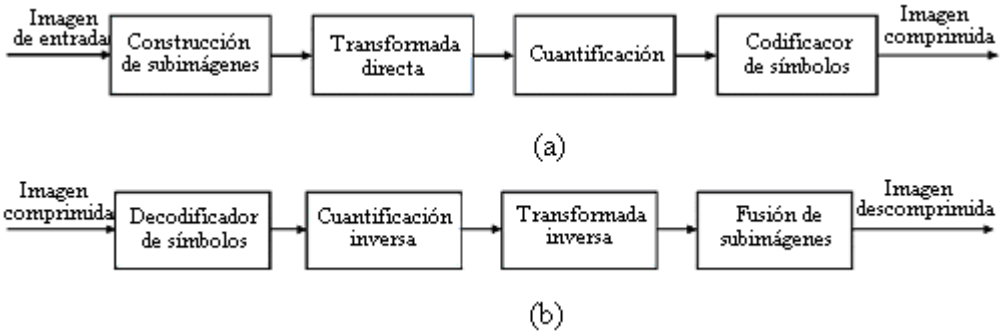

Figura 16. Sistema de compresión por DCT. a) Codificador b) Decodificador

En primer lugar, la imagen a comprimir se divide en bloques o subimágenes de tamaño reducido sobre las que se aplica la DCT. El resultado de la transformación de cada bloque se cuantifica y posteriormente se aplican códigos eficientes para transmitir o almacenar esta información. La decodificación de la imagen se realiza aplicando el proceso inverso: los coeficientes se decodifican y se aplica la transformada inversa, de manera que se recuperan los elementos de imagen en el dominio espacial original [16, 23].

Posteriormente, es necesario recomponer la imagen a partir de los bloques en los que se ha dividido originalmente. Esta recomposición de la imagen suele ser uno de los principales problemas de los métodos de compresión por transformada. En efecto, debido a la cuantificación de los coeficientes transformados, la subimagen se recupera con cierto error respecto a la original, error que es muy crítico si se produce en los límites de las subimágenes, debido a que al fusionar todos los bloques aparecerán cambios de nivel de gris bruscos, conocidos como *artefactos de bloque*, y los bordes de los bloques pueden resultar perceptibles.

El problema es particularmente notorio con algunas transformadas como la de Fourier o Walsh-Hadamard, sin embargo, la DCT presenta excelentes propiedades en la codificación de los contornos de las subimágenes que, de hecho, ha sido uno de los motivos principales por los que se ha elegido esta transformada en casi todos los estándares de codificación.

Ahora, en términos matemáticos, se tiene que la DCT de una secuencia unidimensional $x[n]$ de *N* muestras de longitud se define como [24]:

$$C[k]=\alpha[k]\sum_{n=0}^{N-1}x[n]\cos\left(\frac{(2n+1)k\pi}{2N}\right) \qquad 0\le n\le N-1, \quad o\le k\le N-1 \tag{3.13}$$

donde el coeficiente $\alpha[k]$ viene determinado por:

$$\alpha[k]=\begin{cases}\sqrt{1/N} \ si\, k=0\\ \sqrt{2/N} \ si\, k=1,...,N-1\end{cases} \tag{3.14}$$

Ahora bien, la transformada inversa permite determinar la secuencia original a partir de los coeficientes $C[k]$ mediante el uso de la siguiente ecuación:

$$x[n]=\sum_{k=0}^{N-1}\alpha[k]\cdot C[k]\cdot\cos\left(\frac{(2n+1)k\pi}{2N}\right) \qquad 0\le n\le N-1, \quad o\le k\le N-1 \tag{3.15}$$

Las ecuaciones anteriores tienen cierto parecido con las de la transformada discreta de Fourier, con la salvedad de las expresiones en los índices en la función y sobre todo, en que las exponenciales complejas se sustituyen por funciones coseno. De hecho, la diferencia más importante entre ambas transformadas es que la transformada discreta de Fourier es compleja, mientras que la DCT es real, lo cual supone de entrada, una mejor compactación de la información, puesto que de *N* muestras reales se mantiene un total de *N* coeficientes reales que las representan.

Para el tratamiento de imágenes la DCT debe extenderse a dos dimensiones, para lo cual, la ecuación que se utiliza para la transformada directa es:

$$C[k,l]=\alpha[k]\alpha[l]\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}x[n,m]\cos\left(\frac{(2n+1)k\pi}{2N}\right)\cos\left(\frac{(2m+1)l\pi}{2M}\right) \tag{3.16}$$

en la anterior relación *k* representa las filas y *l* las columnas, donde se supone que el bloque de imagen $x[n,m]$ tiene unas dimensiones de *N* filas por *M* columnas, lo que define el ámbito de validez de los índices *n* y *k* entre *0* y *N-1* y de los índices *m* y *l* entre *0* y *M-1*. Generalmente sólo se trata con bloques cuadrados cuyos tamaños son una potencia de 2.

La transformada inversa puede obtenerse de forma análoga:

$$x[n,m]=\sum_{k=0}^{N-1}\sum_{l=0}^{M-1}\alpha[k]\cdot\alpha[l]\cdot C[k,l]\cdot\cos\left(\frac{(2n+1)k\pi}{2N}\right)\cos\left(\frac{(2m+1)l\pi}{2M}\right) \tag{3.17}$$

La ecuación 3.17 sugiere una interpretación interesante de la transformada coseno, en el sentido que se puede agrupar los factores constantes y las funciones coseno en funciones genéricas para expresar la imagen como:

$$x[n,m]=\sum_{k=0}^{N-1}\sum_{l=0}^{M-1}C[k,l]\cdot H_{k,l}[n,m] \tag{3.18}$$

donde:

$$H_{k,l}[n,m]=\alpha[k]\alpha[l]\cos\left(\frac{(2n+1)k\pi}{2N}\right)\cos\left(\frac{(2m+1)l\pi}{2M}\right) \tag{3.19}$$

y pueden interpretarse como una base de subimágenes de tamaño $N\times M$ cuya forma depende de los valores *k* y *l*. Desde este punto de vista, los coeficientes transformados $C[k,l]$ indican en qué medida participa cada una de estas subimágenes en la obtención de la subimagen original. Por tanto, los bloques originales pueden considerarse como una superposición de estas imágenes base donde los coeficientes transformados indican el peso de la imagen base en la reconstrucción de la imagen final. Los coeficientes próximos a cero

pueden, por tanto, eliminarse sin que afecten a la calidad de la imagen reconstruida, ya que su matriz asociada representa poca contribución en la original.

Tal y como se desprende de la ecuación 3.19, las imágenes base están formadas por productos de dos funciones cosenoidales cuyas frecuencias están directamente relacionadas con las variables *k* y *l*. La variable *k* actúa directamente sobre el eje vertical (componente *n* de la imagen) mientras que la *l* lo hace sobre el eje horizontal. A medida que *k* o *l* aumentan, aumenta la frecuencia de la función coseno, por lo que se produce un mayor número de oscilaciones. En consecuencia, los valores de *k* y *l* pequeños representan zonas de baja frecuencia mientras que los valores altos representan alta frecuencia.

Las imágenes base obtenidas para todos los pares de valores *k* y *l* con DCT de 8x8 muestras se representan en la figura 17. En la primera fila de imágenes base (*k=0*) se observa que la frecuencia horizontal de la señal va aumentando a medida que aumenta el valor de *l*. En cambio, si se mantiene *l=0* y desplazándose en el sentido vertical, aumenta la frecuencia vertical manteniendo la imagen constante en el eje horizontal. Las zonas donde tanto *l* como *k* son elevados corresponden a zonas de alta frecuencia en los dos sentidos [16].

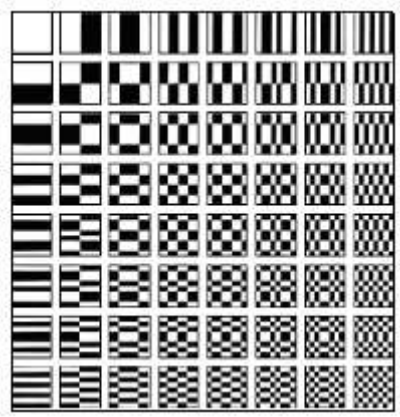

Figura 17. Representación de las imágenes base de la transformada coseno 2D de 8x8

El resultado de una transformada puede interpretarse directamente a partir del conjunto de las imágenes base. Los valores numéricos que se obtienen en la transformada de un bloque de imagen indican el grado en que interviene cada imagen base en la reconstrucción del bloque de imagen original, ahora bien, la posición de las imágenes base es la misma que la

de los coeficientes en la transformada, por lo que los resultados de la transformada pueden proporcionar una idea inmediata del contenido frecuencial de la señal. Adicionalmente, el hecho de que la posición de los coeficientes transformados esté directamente relacionada con el contenido frecuencial de la imagen base, puede ser aprovechado para mejorar la compresión, basándose en las características del sistema visual.

- **Compresión basada en la *transformada wavelet***

La compresión *wavelet* emplea como herramienta matemática la DWT (d*iscrete wavelet transform*) [25, 26] y se basa en la aplicación repetida de filtros paso-alto y paso-bajo sobre una imagen de entrada tanto a filas como a columnas. El filtro paso-bajo extrae la estructura de la señal, mientras que el filtro paso-alto extrae los detalles. Los cuatro filtrados realizados (a filas y a columnas, de paso-bajo y alto), suponen un nivel de la *transformada wavelet* y cada una de las subimágenes obtenidas es llamada *subbanda* [27]. El proceso de filtrado se repite sobre aquella subbanda obtenida luego de aplicar el filtro paso-bajo a filas y columnas, la cual es conocida como *subbanda LL* o *subbanda wavelet*, puesto que ésta contiene la mayor parte de la información. La figura 18 muestra la notación habitualmente utilizada en la descomposición wavelet 2D.

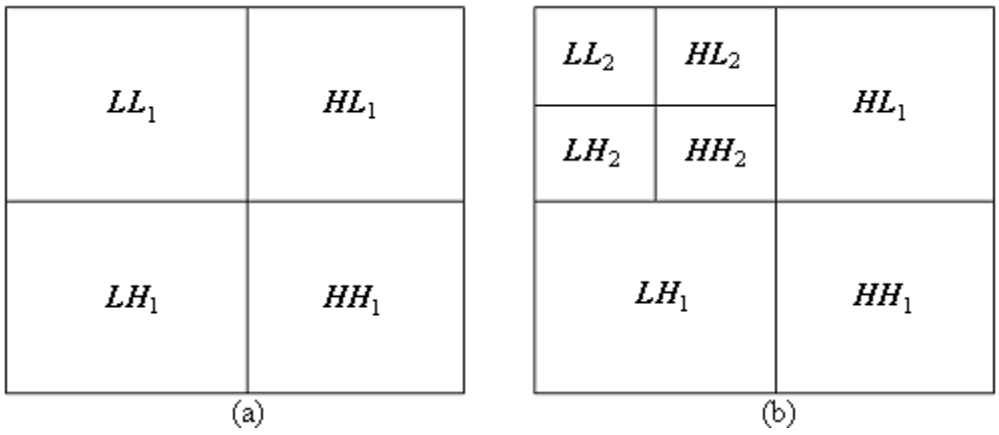

Figura 18. Notación empleada en la descomposición wavelet 2-D. a) de orden 1. b) de orden2

Una descomposición *wavelet* completa se obtiene realizando los pasos de filtrado paso bajo y alto (filtrado y diezmado) sobre la *subbanda wavelet* un número de veces que depende directamente del tamaño de los datos, es decir el proceso se aplica de manera recursiva sobre la banda *LL*. Al final del proceso, se obtiene un conjunto de subbandas, donde sólo una contiene la información relevante de la imagen y el resto contiene los detalles.

En términos matemáticos, si una imagen es sometida al proceso de la DWT se obtiene cuatro tipos de coeficientes: *aproximación* (banda *LL*), *detalles horizontales* (banda *HL*), *detalles verticales* (banda *LH*) y *detalles diagonales* (banda *HH*), donde la aproximación contiene la mayor parte de la energía de la imagen, es decir, la información más importante, mientras que los detalles tienen valores próximos a cero.

La propiedad más importante de la compresión basada en *wavelet* es la eficiente representación de la imagen que se consigue con la aplicación de la DWT, lo cual, constituye la base de la compresión de alta calidad. Otra interesante ventaja de esta transformada frente a otras transformadas como la DCT, es la eliminación de los efectos de bloque producidos en la reconstrucción de la imagen a causa de la aplicación de la transformada sobre bloques de la imagen de entrada, pues la *transformada wavelet* no se aplica sobre bloques sino sobre la imagen en su totalidad.

El propósito ahora es entender la forma cómo opera la *transformada wavelet* y en particular la *wavelet* Haar en una función (señal). Considere una función con 8 muestras, $f \in \mathbb{R}^8$ (no obstante, el procedimiento puede ser generalizado para cualquier señal finita) con $f = (2,5,8,9,7,4,-1,1)$, el objetivo es expandir esta señal según la expresión (2.5.7) de la sección 2.4.3 del capítulo 2, es decir, se desea expandir a f en $V_0 \oplus W_0 \oplus W_1 \oplus W_2$. Basados en la teoría mostrada en la sección 2.4 del capítulo anterior, esta expansión está dada por:

$$\begin{aligned} f = &\langle f, \phi_{0,0} \rangle \phi_{0,0} + \langle f, \psi_{0,0} \rangle \psi_{0,0} + \langle f, \psi_{1,0} \rangle \psi_{1,0} + \langle f, \psi_{1,1} \rangle \psi_{1,1} + \\ &+ \langle f, \psi_{2,0} \rangle \psi_{2,0} + \langle f, \psi_{2,1} \rangle \psi_{2,1} + \langle f, \psi_{2,2} \rangle \psi_{2,2} + \langle f, \psi_{2,3} \rangle \psi_{2,3} \end{aligned} \tag{3.20}$$

El paso a seguir es calcular los productos internos $\langle f,\phi\rangle$ y $\langle f,\psi\rangle$. En principio se podría realizar esto evaluando una integral (o una suma en caso discreto) tal y como se mostró en el capítulo 2, sin embargo en la práctica el cálculo de estos productos internos para una *base wavelet* se hace como se muestra a continuación:

Para seguir con el ejemplo en consideración, se pretende expandir esta señal en la base *Haar* de acuerdo a la expresión (3.20). Realizando convolución discreta entre la señal original y los filtros pasa alto h_1 de coeficientes $\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right)$ y pasa bajo h de coeficientes $\left(-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right)$ hallados en la sección 2.4 del capítulo 2, se obtiene:

Paso 1

$$\begin{aligned} f &= \left(2+5,8+9,7+4,-1+1,2-5,8-9,7-4,-1-1\right)\Big/\sqrt{2} = \\ &= \left(7,17,11,0,-3,-1,3,-2\right)\Big/\sqrt{2} \end{aligned} \tag{3.21}$$

Paso 2

$$\begin{aligned} f &= \left(\frac{7+17}{\sqrt{2}},\frac{11+0}{\sqrt{2}},\frac{7-17}{\sqrt{2}},\frac{11-0}{\sqrt{2}},-3,-1,3,-2\right)\Big/\sqrt{2} = \\ &= \left(\frac{24}{\sqrt{2}},\frac{11}{\sqrt{2}},\frac{-10}{\sqrt{2}},\frac{11}{\sqrt{2}},-3,-1,3,-2\right)\Big/\sqrt{2} \end{aligned} \tag{3.22}$$

Paso 3

$$\begin{aligned} f &= \left(\frac{24+11}{\left(\sqrt{2}\right)^2},\frac{24-11}{\left(\sqrt{2}\right)^2},\frac{-10}{\sqrt{2}},\frac{11}{\sqrt{2}},-3,-1,3,-2\right)\Big/\sqrt{2} = \\ &= \left(\frac{35}{2},\frac{13}{2},\frac{-10}{\sqrt{2}},\frac{11}{\sqrt{2}},-3,-1,3,-2\right)\Big/\sqrt{2} \\ &\approx \left(12.4,4.60,-5.00,5.50,-2.12,-0.707,2.12,-1.41\right) \end{aligned} \tag{3.23}$$

El procedimiento desarrollado anteriormente muestra el proceso de codificar una señal mediante una *transformada wavelet Haar* unidimensional. El interés ahora es saber cuál es

el mecanismo a seguir cuando se trata de señales bidimensionales (imágenes). Pues bien, sencillamente lo que se hace en este caso es actuar con la transformada unidimensional sobre cada fila de la imagen seguida por una transformación de cada columna de la misma.

No obstante, el procedimiento ilustrado para llegar finalmente a la relación (3.23) es claro y fácil de implementar, no muestra en realidad una relación evidente entre esta técnica y la expansión para f expresada en la relación (3.20). Con el propósito de ilustrar que efectivamente esta relación existe, se desarrolla a continuación los mismos pasos anteriores explicando con claridad la correspondencia con los espacios de escalamiento y los *espacios wavelets*, estudiados con detalle en la sección 2.4 del capítulo 2.

Considere la misma función $f = (2,5,8,9,7,4,-1,1)$, los coeficientes de la señal original son los coeficientes del vector f representado en V_3, en otras palabras se tiene: $\langle f, \phi_{3,0} \rangle = 2, \langle f, \phi_{3,1} \rangle = 5, \langle f, \phi_{3,2} \rangle = 8, \ldots \langle f, \phi_{3,7} \rangle = 1$.

Es importante recordar que los vectores de la base de V_3 son traslación de la función escala madre *Haar* la cual ha sido dilatada para que cada función base tenga un soporte igual a 1/8 (= 1 píxel) del soporte de f que es 8 píxeles. El objetivo ahora es expandir f en $V_0 \oplus W_0 \oplus W_1 \oplus W_2$, para lo cual se desarrolla los siguientes pasos.

Paso 1

Lo primero que hay que hacer es expandir f en $V_2 \oplus W_2$. El primer paso en el ejemplo anterior se puede describir como la multiplicación $f_1 = W_1 f$, donde:

$$W_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \tag{3.24}$$

Note que las primera 4 filas corresponden a los vectores base $\phi_{2,0}$, $\phi_{2,1}$, $\phi_{2,2}$ y $\phi_{2,3}$ respectivamente, los cuales generan V_2 y las últimas 4 filas corresponden a los vectores base $\psi_{2,0}$, $\psi_{2,1}$, $\psi_{2,2}$ y $\psi_{2,3}$ los cuales generan W_2.

Ahora, el vector final en el paso 1 del ejemplo anterior no es más que los coeficientes de f en la expansión (3.25).

$$f = \frac{1}{\sqrt{2}}\left(7\phi_{2,0} + 17\phi_{2,1} + 11\phi_{2,2} + 0\phi_{2,3} - 3\psi_{2,0} - 1\psi_{2,1} + 3\psi_{2,2} - 2\psi_{2,3}\right)$$

(3.25)

Es decir, se tiene expandida la señal en $V_2 \oplus W_2$.

Paso 2

Ahora se debe expandir f en $V_1 \oplus W_1 \oplus W_2$. En este paso es fácil notar que ya se tiene los coeficientes para los vectores base de W_2, por lo tanto, simplemente se mantiene las cuatro últimas entradas del vector en el paso previo y solo se trabaja con las cuatro primeras. Este paso puede describirse como la multiplicación $f_2 = W_2 f_1$, donde:

$$W_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.26}$$

Se puede combinar el primero y segundo paso como $f_2 = W_2 W_1 f$, donde:

$$W_2 W_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{3.27}$$

Note que las primeras 2 filas corresponden a los vectores base $\phi_{1,0}$ y $\phi_{1,1}$ los cuales expanden V_1, la tercera y cuarta fila representan a los vectores base $\psi_{1,0}$ y $\psi_{1,1}$ los cuales generan W_1, mientras que las últimas cuatro corresponden a los vectores base de W_2.

Ahora, el vector final en el paso 2 en el ejemplo anterior no es más que los coeficientes de f en la expansión:

$$f = \frac{1}{\sqrt{2}}\left(\frac{24}{\sqrt{2}}\phi_{1,0} + \frac{11}{\sqrt{2}}\phi_{1,1} - \frac{10}{\sqrt{2}}\psi_{1,0} + \frac{11}{\sqrt{2}}\psi_{1,1} - 3\psi_{2,0} - 1\psi_{2,1} + 3\psi_{2,2} - 2\psi_{2,3}\right) \qquad (3.28)$$

Paso 3

Igualmente al paso previo, se observa que ya se tiene los coeficientes para los vectores base de $W_1 \oplus W_2$, por lo tanto, se mantiene las últimas seis entradas del vector en el paso anterior y solo se trabaja con las primeras dos. El paso 3 puede describirse como la multiplicación $f_3 = W_3 f_2$, donde:

$$W_3 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.29)$$

Se puede obviamente combinar el primero, segundo y tercer paso como: $f_3 = W_3 W_2 W_1 f$, donde:

$$W_3W_2W_1 = \begin{pmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \qquad (3.30)$$

De nuevo se observa que la primera fila corresponde al vector base $\phi_{0,0}$ que genera V_0 y la segunda representa al vector base $\psi_{0,0}$ que genera W_0, así sucesivamente.

Por último, el vector obtenido en el paso 3 no es más que los coeficientes de f en la expansión:

$$f = \frac{1}{\sqrt{2}}\left(\frac{35}{2}\phi_{0,0} + \frac{13}{2}\psi_{0,0} - \frac{10}{\sqrt{2}}\psi_{1,0} + \frac{11}{\sqrt{2}}\psi_{1,1} - 3\psi_{2,0} - 1\psi_{2,1} + 3\psi_{2,2} - 2\psi_{2,3}\right) \quad (3.31)$$
$$\approx \left(12.4\phi_{0,0} + 4.60\psi_{0,0} - 5.00\psi_{1,0} + 5.50\psi_{1,1} - 2.12\psi_{2,0} - 0.707\psi_{2,1} + 2.12\psi_{2,2} - 1.41\psi_{2,3}\right)$$

Finalmente el ejemplo anterior puede ser expresado como:

$$Wf = \frac{1}{\sqrt{2}}\left(\frac{35}{2}\phi_{0,0} + \frac{13}{2}\psi_{0,0} - \frac{10}{\sqrt{2}}\psi_{1,0} + \frac{11}{\sqrt{2}}\psi_{1,1} - 3\psi_{2,0} - 1\psi_{2,1} + 3\psi_{2,2} - 2\psi_{2,3}\right). \quad (3.32)$$

Los procesos descritos anteriormente pueden representarse mediante el esquema mostrado en la figura 3.19, el cual es una extensión del caso unidimensional al caso bidimensional y generaliza el empleo de cualquier *wavelet* a través de sus filtros de descomposición pasa bajo y pasa alto, Lo_D, Hi_D respectivamente [28].

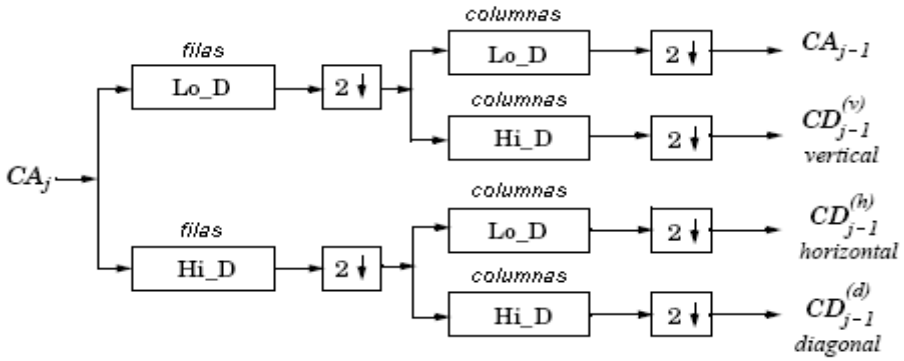

Figura 19. Esquema de descomposición 2-D

Análogamente, para el proceso de reconstrucción se emplea el esquema mostrado en la figura 20 donde los filtros de reconstrucción Lo_R y Hi_R, pasa bajo y pasa alto respectivamente, dependen de la *wavelet* empleada.

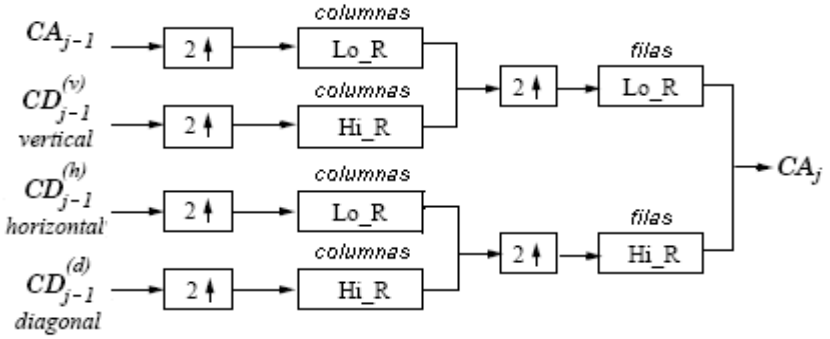

Figura 20. Esquema de reconstrucción 2-D

No obstante, el especial interés que en los últimos ha generado la aplicación de la teoría *wavelet* para la compresión de imágenes, ha dado como resultado importantes avances orientados a obtener mayor provecho de sus propiedades. Es precisamente el concepto de codificación de *árboles de ceros*, tratado en la siguiente sección, uno de los avances más notorios, a tal punto de marcar el inicio de la denominada era de los sistema de *compresión wavelet* modernos.

3.3.3 Codificación mediante árboles de ceros embebidos (EZW)

Como ya se mencionó, una de las propiedades interesantes de la *transformada wavelet*, relativa a la compresión, es la compactación de la energía. Debido a esta propiedad, los diseñadores de sistemas de codificación encontraron que se pueden conseguir factores de compresión altos y un error cuadrático medio (MSE) pequeño codificando solamente los pocos coeficientes con mayor energía. El problema es que al codificar un número pequeño de coeficientes, el codificador debe enviar la información de la posición de éstos, de manera que los datos se puedan decodificar correctamente, Ahora bien, dependiendo del método empleado, los recursos necesarios para codificar la información de la posición pueden ser una fracción significativa, contrarrestando los beneficios de la compactación de la energía.

Durante los últimos años se han propuesto varios métodos para disminuir el número de bits necesarios para codificar la información de la posición asociada a los coeficientes

significativos, muchos de los cuales se basan en la idea de la *predicción interbanda*, la cual saca provecho de la naturaleza jerárquica de la transformada wavelet. La idea de la *predicción interbanda* es usar la posición de los coeficientes significativos en una subbanda para tratar de predecir la posición y la magnitud de los coeficientes significativos en las otras subbandas (a otro nivel de descomposición), reduciendo de este modo el coste de codificar la información de la posición [29].

El algoritmo EZW (*Embedded zerotree wavelet*) propuesto por Jerry Shapiro en 1993, reconoce que una fracción importante del total de bits requeridos para codificar una imagen se necesita para codificar la información de la posición, o lo que el EZW denomina *mapas de significancia.* Un *mapa de significancia* se define como un indicador de cuándo un coeficiente particular es *cero* (no importante) o *no-cero* (importante) respecto a un nivel de cuantificación dado. El algoritmo EZW determina en esencia una manera muy eficiente de codificar los mapas de significancia codificando la localización de los *ceros* [30].

El EZW se fundamenta en el concepto de *árbol de ceros* o *zerotree*, el cual está basado en la hipótesis que si un coeficiente wavelet de una escala mayor es insignificante respecto a un umbral dado T, entonces todos los coeficientes de la misma orientación espacial en escalas más finas, son probablemente también insignificantes respecto a T. La relación entre coeficientes de distintas subbandas es como se presenta en la figura 21.

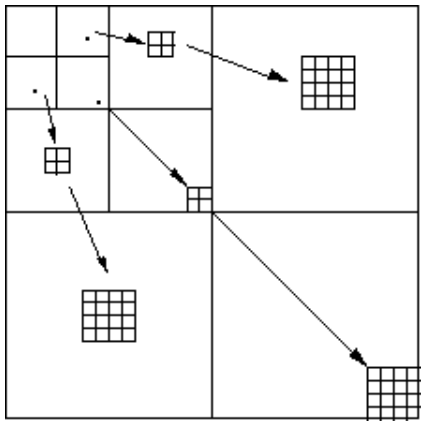

Figura 21. Relación *padre – hijos* entre coeficientes de distintas subbandas

Al conjunto de coeficientes insignificantes respecto al umbral, se les denomina árbol de ceros. Estos árboles son una técnica muy potente de codificación puesto que declarando un solo coeficiente como raíz de un árbol de ceros, automáticamente se sabe que un gran número de coeficientes son cero, además, la representación compacta de dichos árboles y el hecho de que éstos ocurren frecuentemente, especialmente para factores de compresión altos, hacen que esta técnica sea muy eficiente para codificar la información de la posición.

Otro aspecto importante del algoritmo EZW es el concepto de cuantificación por aproximaciones sucesivas, cuya idea básica es la disminución gradual del MSE a medida que se detecta *coeficientes significativos* como consecuencia de un recorrido iterativo de los coeficientes *wavelet* usando sucesivamente umbrales decrecientes. Este tipo de cuantificación es similar a la representación binaria de un número real, en la que cada bit que se agrega a la representación del número, añade más precisión.

La cuantificación por aproximaciones sucesivas cumple dos tareas específicas en el algoritmo EZW: primero, este método genera un gran número de árboles de ceros, que resultan fáciles de codificar, en segundo lugar, ordena los bits codificados de manera que los más significativos se envían primero. Esto último resulta ser de gran importancia puesto que genera un código *embebido,* lo cual significa que los bits necesarios para representar una imagen a una resolución alta se pueden derivar simplemente añadiendo bits de refinamiento a la representación de baja resolución de la imagen. De manera equivalente, una versión de baja resolución se puede conseguir simplemente truncando la cadena de bits embebida, dando como resultado una tasa binaria media inferior [30].

3.4 CRITERIOS DE CALIDAD O FIDELIDAD

En el proceso de compresión de imágenes con pérdidas, tal como su nombre lo indica, se presenta una pérdida real o cuantitativa de información visual, por lo tanto se requiere de métodos que permitan evaluar esas pérdidas, es decir, cuantificar la naturaleza y el alcance

de la pérdida de la información [31]. Esa valoración se puede realizar teniendo en cuenta los criterios de calidad o fidelidad, éstos se clasifican en:

- Criterios de calidad objetivos
- Criterios de calidad subjetivos

Los primeros se emplean cuando es posible expresar el nivel de pérdida de información como una función de la imagen de entrada y de la imagen de salida comprimida que posteriormente se reconstruye. Un ejemplo de ello es el error cuadrático medio (MSE) [32], el cual se define a continuación.

Suponga que se tiene una función $f(x,y)$ la cual representa una imagen de entrada de tamaño $M \times N$, y suponga también que $f'(x,y)$ representa una aproximación de $f(x,y)$ producto de la compresión y posterior reconstrucción de la imagen de entrada. Luego, el error cometido entre las dos imágenes durante este proceso denotado por $e(x,y)$ se define como:

$$e(x,y) = f'(x,y) - f(x,y). \tag{3.33}$$

El error total entre las dos imágenes medido como error cuadrático medio *MSE* se define como:

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f'(x,y) - f(x,y)\right]^2, \tag{3.34}$$

ahora bien, el error cuadrático medio E_{rms} entre $f(x,y)$ y $f'(x,y)$ es:

$$E_{rms} = \left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f'(x,y) - f(x,y)\right]^2\right]^{1/2}. \tag{3.35}$$

Otro criterio de fidelidad objetiva estrechamente relacionado a los anteriores es la media cuadrática de la relación señal a ruido, la cual se representa como *SNR* y que se define así:

$$SNR = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f'(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f'(x,y) - f(x,y)\right]^2}, \tag{3.36}$$

ahora, el valor cuadrático medio (*rms*) de la relación señal a ruido, representado por SNR_{rms}, se obtiene tomando la raíz cuadrada de la ecuación anterior.

Otro criterio de calidad objetivo es la *PSNR* (*Pick Signal to Noise Ratio*), definida como:

$$PSNR = 10 * \log_{10}\left(\frac{255^2}{MSE}\right) \tag{3.37}$$

Los criterios anteriormente expuestos [18], como su nombre lo indica ofrecen una alternativa objetiva para evaluar la pérdida de información, pero como las imágenes reconstruidas son vistas finalmente por el ojo humano, es apropiado hacer evaluaciones basadas en criterios de calidad subjetivos. Ello se consigue mostrando una imagen reconstruida a un grupo adecuado de personas para después promediar las evaluaciones y concluir subjetivamente sobre la calidad de la misma. Dicha calidad puede medirse bajo una escala de criterios preestablecidos, como por ejemplo, buena, regular, mala y pésima, o también, excelente, buena, aceptable y no utilizable. De cualquier modo, lo importante es definir una escala cualitativa para medir la bondad de la imagen reconstruida.

4 ALGORITMO DE COMPRESIÓN Y RECONSTRUCCIÓN DE IMÁGENES CON *WAVELETS*

En este capítulo se presenta las consideraciones tenidas en cuenta en los procesos de análisis, diseño e implementación para el desarrollo del algoritmo de compresión y reconstrucción de imágenes. Se describe sus características, alcance, limitaciones y modo de operación.

4.1 FASE DE ANÁLISIS

Para la implementación del algoritmo se toma como punto de partida la información de los capítulos previos. De este modo, la teoría matemática básica presentada en el capítulo 2, así como el referente teórico sobre la compresión de imágenes abordado en el capítulo 3, hacen parte del estudio y análisis requerido para el desarrollo del mismo.

Las consideraciones básicas o requerimientos identificados como paso previo al diseño e implementación del algoritmo son:

- Desarrollar un sistema compresor con pérdidas de imágenes fijas en escala de grises mediante codificación por transformada, aplicando la *transformada wavelet*. El algoritmo debe constar de unos componentes básicos o mínimos de un sistema compresor de imágenes como los identificados en el capítulo 3. (Ver figura 13)

- Establecer un entorno de experimentación en el campo de la compresión de imágenes con *wavelets*, experimentando con varias *funciones wavelets* entre las cuales se considera la más simple de ellas que es la *Haar*.
- Presentar al usuario los resultados de la aplicación de la *transformada wavelet* en la compresión de imágenes a través de una interfaz gráfica, no sólo con el propósito de evaluar objetiva y subjetivamente a la imagen reconstruida, sino además visualizar otros efectos de la utilización de la transformada.

4.2 FASE DE DISEÑO

Teniendo en cuenta que la herramienta software empleada para el desarrollo del algoritmo es Matlab, que cuenta con un toolbox especializado para la aplicación de la *teoría wavelet*, es importante mencionar que para su implementación se hace uso de algunas funciones propias de este toolbox, obviamente complementadas con funciones desarrolladas en forma adecuada para cada módulo que conforma un esquema de compresión de imágenes.

4.2.1 *Toolbox wavelet*

La *transformada wavelet* poco a poco tiende a ser aceptada como una herramienta útil para diferentes aplicaciones, pues su período de presentación está culminando para dar paso a su etapa de consolidación e implementación en diferentes campos. En éste contexto puede entonces catalogarse el *toolbox wavelet* [4] como una herramienta no sólo útil sino también necesaria, que proporciona un entorno de trabajo propicio para la experimentación e implementación de diferentes aplicaciones. Puesto que existen diversos campos de aplicación de la *teoría wavelet*, el toolbox cuenta con una gran variedad de utilidades, entre las cuales se destacan: herramientas de análisis y síntesis, procesamiento de imagen y señal, utilidades para eliminación de ruido, compresión de señales unidimensionales y bidimensionales, etc.

El toolbox incluye dos modos de operación, uno por medio de “línea de comandos” y otro por “interfaz gráfica de usuario”, ambos con la capacidad de complementarse uno al otro. La interfaz gráfica es una gran herramienta tanto para quienes apenas comienzan a interesarse en la *teoría wavelet*, como para los expertos en el tema permitiendo la realización de pruebas; mientras que por línea de comandos se da un entorno abierto para la experimentación a través de las funciones que el toolbox ofrece.

Definitivamente el toolbox *wavelet* se constituye en una gran alternativa para trabajar con *wavelets*, pues junto a la potencialidad propia de Matlab, permite realizar complejas y poderosas aplicaciones.

4.2.2 Entorno de desarrollo de interfaces gráficas de usuario

Puesto que el volumen y la complejidad de los datos crecen con el incremento de la complejidad de las fuentes de datos y algoritmos, la necesidad de representaciones intuitivas de esos datos y resultados se vuelve cada vez más crítica. La representación gráfica es a menudo no solo el medio más efectivo de transmitir los puntos de estudio o trabajo que han proporcionado dichos datos, sino también una expectativa de la audiencia del trabajo. Matlab sigue siendo entonces una de las mejores aplicaciones disponibles para brindar la capacidad computacional tanto de generar datos, como de desplegarlos en una variedad de representaciones gráficas.

El entorno de desarrollo de interfaces gráficas de usuario en Matlab proporciona un conjunto de herramientas para crear GUI’s (interfaces gráficas de usuario) [33], las cuales simplifican considerablemente el proceso de diseño y programación de una GUI a través de línea de comandos en el editor de Matlab.

Para hacer uso de el entorno de desarrollo de GUI’s en Matlab (GUIDE) se debe ejecutar el comando *guide* en el prompt, e inmediatamente se despliega el denominado *Layout Editor* o editor de diseño, el cual es el panel de control para todas las herramientas GUIDE.

El editor de diseño permite crear rápida y fácilmente una GUI arrastrando componentes gráficos tales como: botones, menús, cajas de texto, etiquetas o ejes, desde una paleta de componentes al área de diseño.

Cuando se culmina el diseño de la interfaz y se guarda, se genera un archivo *.fig* que contiene información sobre las propiedades de los componentes gráficos empleados en la interfaz; así mismo, es generado un archivo *.m* con el mismo nombre del archivo *.fig*, donde se da la posibilidad al usuario de programar el comportamiento de los componentes al ser empleados (*callbacks*), y donde además es posible configurar las propiedades de los mismos. De hecho, en muchos casos se hace necesario adicionar o configurar componentes directamente sobre el archivo con extensión *.m*, generado por GUIDE, haciendo uso del editor de Matlab.

Considerando la gran funcionalidad proporcionada por el *toolbox wavelet* de Matlab y la facilidad de representación gráfica brindada por el entorno GUIDE, el algoritmo desarrollado aprovecha ambas utilidades. Así pues, el algoritmo emplea algunas de las funciones del toolbox para el procesamiento matemático requerido en la compresión y reconstrucción de imágenes con *wavelets*, mientras que, a través del entorno gráfico brinda acceso a toda su funcionalidad de una manera sencilla, mostrando los resultados obtenidos.

4.2.3 Diagrama en bloques del sistema

El sistema compresor de imágenes desarrollado consta de los módulos presentados en la figura 13 del capítulo 3 con algunas adiciones detalladas posteriormente. A continuación se nombra la técnica utilizada en cada uno de ellos para su posterior análisis.

En la etapa de compresión de energía se aplica directamente la DWT a los datos de la imagen, luego se implementa una etapa de umbralización que permite eliminar los coeficientes menos significativos y puesto que es deseable que el sistema produzca una cadena de código embebida, de manera que a medida que se lea la cadena, se añada detalle

a la imagen reconstruida, los coeficientes resultantes son cuantificados mediante el algoritmo EZW. Como esquema de codificación de entropía se implementa un codificador aritmético, para finalmente obtener una cadena de bits (*bitstream*) de salida que se almacenan en un archivo (o fichero) que representa la imagen comprimida. En el decodificador, primero se decodifica la cadena de código mediante el decodificador aritmético, a continuación, se decuantifican los coeficientes transformados mediante el decodificador EZW, y finalmente se realiza la transformada inversa, dando como resultado la imagen reconstruida.

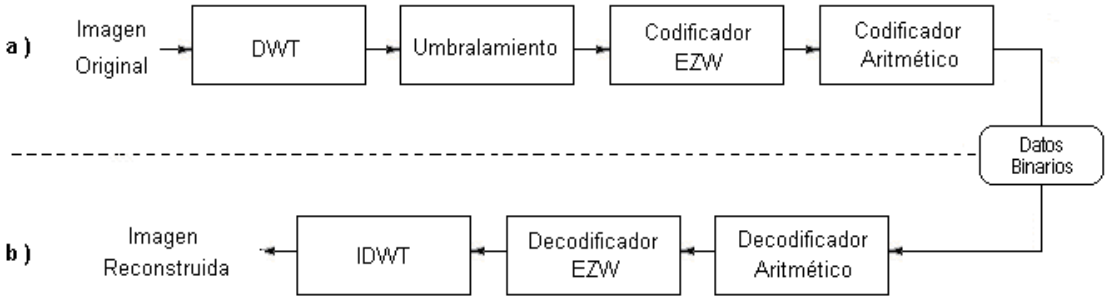

Figura 22. Sistema implementado a) Codificador, b) Decodificador

El diagrama final diseñado para el sistema compresor de imágenes se muestra en la figura 22. A continuación se presenta las consideraciones tenidas en cuenta en cada una de las etapas del codificador y del decodificador para el diseño del sistema.

4.2.4 Codificador

4.2.4.1 DWT o descomposición subbanda

El algoritmo permite realizar la denominada *descomposición subbanda* a una imagen previamente cargada el la GUI de algoritmo, lo cual hace referencia a la representación gráfica de los coeficientes resultantes de la *transformada wavelet* de la imagen. Tales coeficientes se diferencian como: aproximación y como detalles, estos últimos de tres tipos:

horizontales, verticales y diagonales, además, dicha descomposición puede iterarse sobre la imagen de aproximación resultante de cada paso de descomposición, logrando así tantos niveles de *análisis wavelet* como sea posible. Como se aprecia, es en esta etapa del procesamiento donde se realiza la DWT como tal sobre la imagen.

4.2.4.2 Umbralización

La segunda etapa del algoritmo es la de umbralización, donde se elimina cierta cantidad de *coeficientes wavelets* de detalle, aquellos que sean menores a un umbral elegido. Solamente los coeficientes de detalle son sometidos a umbralización partiendo del hecho que la mayor cantidad de energía de una imagen (o de una señal en general) se concentra en las bajas frecuencias, en otras palabras, en los coeficientes de aproximación. Considerando que se cuenta con la descomposición subbanda o multinivel realizada en la etapa previa, el algoritmo brinda la posibilidad de realizar umbralización global o umbralización por nivel.

La umbralización global o umbralización universal, tal como su nombre lo indica, permite eliminar coeficientes (asignarles valor cero) de detalle en todas las subbandas y para todos los niveles al tiempo, empleando un único umbral (claro está, exceptuando la subbanda LL de último nivel). Otra alternativa es la umbralización por nivel, donde se puede establecer diferentes umbrales para cada una de las subbandas HL, LH y HH a todos los niveles considerados en la descomposición.

4.2.4.3 Codificador EZW

La implementación desarrollada emplea el algoritmo EZW como mecanismo de cuantificación, que a su vez facilita la codificación entrópica implementada en el siguiente módulo. Como ya se mencionó, la cuantificación de los coeficientes implica una pérdida de información que hace que la imagen decodificada no sea idéntica a la imagen original,

pero a cambio de ello, permite reducir el número de bits necesarios para codificar la imagen.

La implementación del EZW se realiza mediante un algoritmo conocido como *algoritmo de incrustación*, el cual cuantifica los *coeficientes wavelet* de la imagen en pasos dados por potencias de dos que se denominan niveles EZW, reduciéndose progresivamente en sucesivas iteraciones. El primer nivel EWZ toma como valor 2^{t_0}, donde t_0 es el umbral inicial que depende del valor del máximo coeficiente transformado, mientras que el último nivel EZW es 1 (2^0). Entonces, la primera operación a realizar consiste en el cálculo del umbral de partida t_0 de acuerdo a la siguiente relación:

$$t_0 = 2^{\left\lfloor \log_2\left(MAX\left(|\gamma(x,y)|\right)\right)\right\rfloor} \qquad (4.1)$$

donde $\gamma(x,y)$ representa la imagen transformada.

A continuación, se construye un mapa de significancia o de árboles de ceros basándose en la búsqueda de los coeficientes mayores o iguales al umbral determinado en el paso anterior. Existen diferentes órdenes para recorrer y evaluar los coeficientes, entre los cuales el más difundido es el denominado orden de extracción de Morton representado en la figura 23, además cabe mencionar que es precisamente esta técnica la realizada en esta implementación. Su fundamento radica en que previamente a la extracción de un coeficiente perteneciente a una determinada subbanda, se extrae los coeficientes con la misma orientación espacial pertenecientes a las subbandas inferiores, lo que supone una mayor prioridad a la extracción de los coeficientes más significativos.

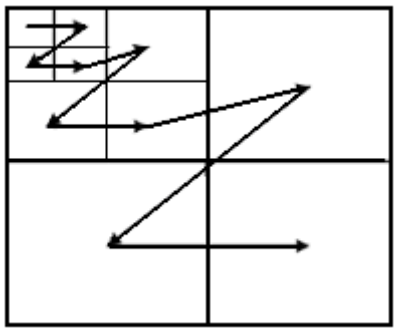

Figura 23. Orden de extracción de Morton

La implementación de la cuantificación por aproximaciones sucesivas mencionada en el capítulo 3, se realiza a través de un recorrido iterativo de los *coeficientes wavelet* usando sucesivamente umbrales decrecientes, cada recorrido de estos coeficientes se divide en dos pasadas: *pasada dominante* y *pasada subordinada*.

La *pasada dominante* establece un mapa de significancia de los *coeficientes wavelet* con respecto al umbral actual, para ello, la imagen transformada es recorrida y se genera un símbolo por cada coeficiente así: si el coeficiente es mayor que el umbral, se codifica un símbolo 'p', si es menor que menos el umbral, se codifica un símbolo 'n', si el coeficiente es la raíz de un árbol de ceros se codifica un símbolo 't', finalmente, si el coeficiente es menor que el umbral pero no es raíz de un árbol de ceros, se codifica un símbolo 'z' (llamado cero aislado).

Es fácil observar que para determinar si un coeficiente es la raíz de un árbol de ceros o un cero aislado, se debe evaluar todo su árbol de descendencia, además, se debe llevar cuenta de todos los elementos pertenecientes a un árbol de ceros para prevenir su codificación posteriormente, lo cual implica tiempo y recursos computacionales para su procesamiento. Finalmente, todos los coeficientes que son mayores en valor absoluto al umbral actual, son extraídos y colocados sin signo en una lista denominada *lista subordinada* y sus posiciones en la imagen son llenadas con cero, lo cual previene que sean codificados nuevamente.

Por otro lado, la denominada *pasada subordinada* es un paso de refinamiento, y es donde se busca el siguiente bit más significativo de los coeficientes que se han determinado significativos en pasadas anteriores.

Luego de hacer la pasada subordinada, se reduce el umbral de análisis a la mitad y se repite el proceso. La codificación EZW termina cuando se alcanza un umbral mínimo preestablecido, de esta manera, la salida del codificador EZW consiste de una lista dominante y una lista subordinada por cada umbral de análisis, la primera compuesta por una secuencia de símbolos 'z', 't', 'n', o 'p' y la segunda por una cadena de bits.

4.2.4.4 Codificador aritmético

De acuerdo a la sección 3.3.1 del capítulo anterior, para el desarrollo de este proyecto en este módulo de codificación de entropía mostrado en la figura 22, se realiza una codificación aritmética con un modelo de datos adaptativo (este modelo de datos es estudiado más adelante). Ahora bien, no obstante la codificación aritmética es más compleja desde el punto de vista matemático, es la que mejores factores de compresión proporciona, adicionalmente, no presenta inconvenientes cuando se combina con un modelo de compresión adaptativo como en el caso de la Huffman.

Como se mencionó en el capítulo 3, este tipo de codificación, toma una secuencia completa de símbolos fuente y la codifica mediante un único número de salida en punto flotante. La salida de un codificador aritmético es un único número, menor que 1 y mayor o igual que 0, para ello es necesario asignar un conjunto de probabilidades a los símbolos que se van a codificar.

El modo de funcionamiento de la codificación aritmética se entiende mejor mediante un ejemplo, así: si se desea codificar el mensaje aleatorio “TESIS”, se tiene una distribución de probabilidad como la mostrada en la tabla 1.

Tabla 1. Distribución de probabilidad para el mensaje “TESIS”

Símbolo	Probabilidad
E	1/5
I	1/5
S	2/5
T	1/5

Ahora se asigna a cada símbolo un rango de acuerdo a su probabilidad dentro del segmento de probabilidad desde cero (0) a uno (1), el orden de esta asignación dentro de este segmento no importa, lo importante es que se haga de la misma manera tanto en el codificador como en el decodificador. En la tabla 2, se muestra esta asignación para el ejemplo en estudio.

Tabla 2. Rango asignado a cada símbolo del mensaje "TESIS"

Símbolo	Probabilidad	Rango
E	1/5	0.00 – 0.20
I	1/5	0.20 – 0.40
S	2/5	0.40 – 0.80
T	1/5	0.80 – 1.00

Ahora bien, la parte más significativa de un mensaje codificado aritméticamente corresponde al primer símbolo codificado, en el ejemplo es el caracter 'T', por lo tanto para que el primer caracter se decodifique correctamente, el mensaje codificado final tiene que ser un número mayor o igual que 0.80, y menor que 1.00. Para elegir este número final, se guarda un registro del rango dentro del cual debe caer el número, luego, tras codificar el símbolo 'T', el extremo inferior de este rango será 0.80, y el extremo superior será 1.00. Durante el resto del proceso de codificación, cada símbolo que sea codificado restringirá aún más el rango del número de salida.

El siguiente símbolo a codificar es el caracter 'E', que posee el rango 0.00 – 0.20, como en este caso el símbolo 'E' es el segundo carácter, lo que se hace es decir que al símbolo 'E' le corresponde el rango 0.00 - 0.20 dentro del subrango 0.80 – 1.00, por tanto, el número codificado estará entre 0.80 y 0.82.

En términos matemáticos, las relaciones utilizadas para actualizar el valor de los extremos de los rangos correspondientes a cada símbolo en codificar según las probabilidades respectivas se expresan a continuación.

$$rango = superior - inferior \tag{4.2}$$

$$superior = inferior + rango * superior(símbolo) \tag{4.3}$$

$$inferior = inferior + rango * inferior(símbolo) \tag{4.4}$$

donde *superior* e *inferior* representan respectivamente los valores de los extremos superior e inferior del rango correspondiente al símbolo respectivo. Con las relaciones anteriores, se realiza el proceso hasta el final del ejemplo, el cual se muestra en la tabla 3.

Tabla 3. Proceso de codificación para el mensaje "TESIS"

Nuevo símbolo	Valor inferior	Valor superior
	0.0	1.0
T	0.8	1.0
E	0.8	0.84
S	0.816	0.832
I	0.8192	0.8224
S	0.82048	0.82176

Por lo tanto, el valor inferior final, 0.82048, codifica de manera unívoca el mensaje "TESIS" aritméticamente.

Modelo de datos

Algo de suma importancia a tener en cuenta es el modelo de datos con el cual se va a trabajar, que corresponde a un modelo de actualización de las estadísticas de los datos que se van codificando. En este proyecto se desarrolla la codificación aritmética la cual necesita precisamente de un modelo de datos estadísticos para realizar la compresión de una forma eficiente, para ello dicho modelo debe cumplir con las siguientes características:

- Predecir la probabilidad de los símbolos en el mensaje de entrada de una manera precisa.
- La distribución de las probabilidades de los símbolos generadas por el modelo no puede ser uniforme.

Como ya se mencionó, el modelo de datos implementado es un modelo adaptativo en el que las probabilidades de los símbolos se actualizan según se codifican los símbolos del mensaje de entrada (en el modelo no adaptativo se realiza una pasada sobre los datos de entrada para obtener las estadísticas de los mismos, después se realiza la codificación de los datos con esas estadísticas las cuales no varían durante todo el proceso). Así mismo, en el proceso inverso, es decir durante la decodificación, las probabilidades se actualizan conforme se decodifican los símbolos.

Dado que el codificador aritmético recibe cadenas de símbolos provenientes del codificador EZW, el modelo de datos necesita almacenar las estadísticas de cinco símbolos ('z', 't', 'n', 'p' y EOF) correspondientes a los cuatro símbolos EZW, y el EOF que representa el símbolo final de mensaje.

En este caso el modelo implementado consiste simplemente en un arreglo de seis posiciones, llamado *modelo*, que se muestra en la figura 24.

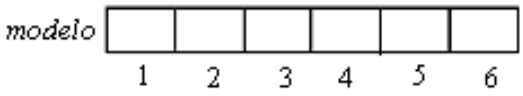

Figura 24. Arreglo implementado como modelo de datos

La correspondencia de las posiciones del arreglo con cada uno de los símbolos es como sigue: La posición uno (1) corresponde al símbolo EOF, la posición dos (2) al símbolo 'z', la tres (3) al símbolo 't', el símbolo 'n' corresponde a la posición cuatro (4) y finalmente la posición cinco (5) al símbolo 'p', la posición seis (6) corresponde al número total de símbolos codificados. Además, el valor *modelo*[*i*] equivale al extremo inferior del rango del símbolo i-ésimo, mientras que el extremo superior se almacena en *modelo*[*i+1*].

Como se mencionó previamente, una característica de un modelo de datos adaptativo es que tanto el codificador como el decodificador inicializan el modelo antes de comenzar, asignando el mismo rango a cada símbolo. Conforme se realiza la codificación o decodificación, se incrementa el rango del símbolo codificado o decodificado.

4.2.4.5 Datos binarios

En las secciones anteriores se ha explicado en detalle cada uno de los módulos que componen el sistema implementado de compresión de imágenes, ahora bien, el procedimiento a seguir es guardar los datos comprimidos en un archivo ya sea para ser

almacenados, transmitidos o reproducidos. Es con este objetivo que después del bloque del codificador aritmético se implementa el módulo de "datos binarios" (ver figura 22).

Una de las principales características que debe cumplir el archivo del cual se hace mención es que a la hora de su transmisión o decodificación se pueda realizar de una manera progresiva, es por ello que las listas de símbolos EZW codificadas aritméticamente se deben guardar ordenadas según su nivel. Por otro lado, es de suma importancia agregar a dicho archivo una cabecera que contenga además de las características de la imagen original la información de sincronismo para indicar al decodificador los parámetros utilizados durante la codificación, necesarios para una correcta decodificación de la imagen.

La información de sincronismo corresponde a ciertos parámetros propios de la codificación de la imagen cada uno de los cuales debe ser escrito en una línea en modo texto y en estricto orden así: tamaño de la imagen original, los datos de la matriz *S* que especifica las dimensiones de las subbandas de la descomposición leídos por filas, tipo de *transformada wavelet* empleada y número de niveles de *descomposición wavelet*.

A continuación de la cabecera, se debe especificar el valor del umbral inicial de la codificación EZW. Una vez anexada la información anterior, se escriben ahora si las listas de símbolos dominantes y subordinadas codificadas aritméticamente según el nivel EZW.

En la figura 25 se muestra un ejemplo de la estructura de un archivo comprimido.

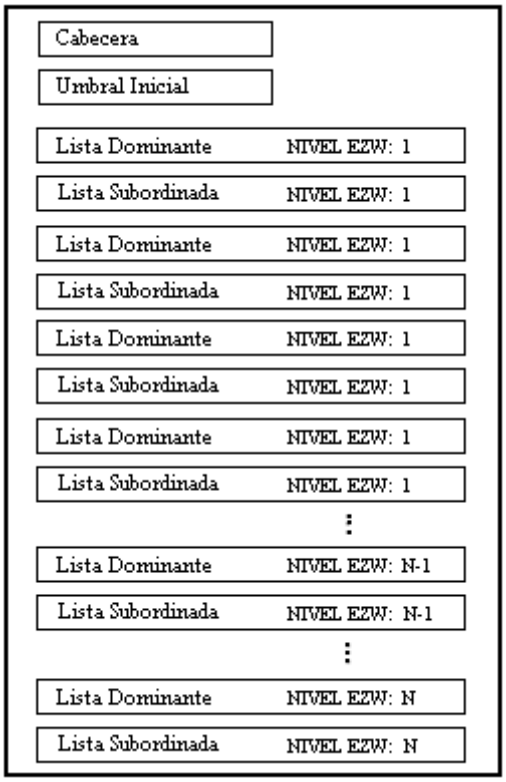

Figura 25. Estructura del archivo comprimido

Ahora, recordando de la sección 4.2.4.4, al codificar aritméticamente las listas de símbolos EZW, se obtiene cadenas de bits de longitud variable, pues bien, para escribir en el archivo estas cadenas, se agrupan estos bits en grupos de ocho y en el archivo comprimido se escribe el código ASCII correspondiente al entero que represente ese grupo de ocho bits. Es importante aclarar que si el tamaño en bits de una cadena en particular no es múltiplo de ocho, se añaden los ceros necesarios al último grupo y se procede como se indicó, por otro lado es fácil deducir que esta adición de ceros no afecta a la codificación aritmética puesto que el valor del número al que representa esa cadena en particular no varía.

Por otro lado, se debe implementar un método para indicar al decodificador dónde empieza y dónde acaba cada una de las listas EZW, el método aquí utilizado es la llamada *técnica de caracteres de relleno*, que consiste en comenzar cada lista con la secuencia de caracteres ASCII “DLE” “STX”, y terminarla con la secuencia “DLE” “ETX”, así, el decodificador sabrá distinguir exactamente dónde empieza y dónde termina cada lista.

Ahora bien, debido a que las listas se componen de datos binarios agrupados en grupos de 8 bits, las secuencias de caracteres “DLE” “STX” o “DLE” “ETX” pueden ocurrir en los datos, produciendo que el decodificador confunda esta información con el inicio o final de una lista. Una manera de solucionar este inconveniente consiste en añadir un carácter ASCII “DLE” justo antes de cada carácter “DLE” *accidental* en el código al escribir el archivo, es decir, los símbolos “DLE” en los datos aparecerían siempre de dos en dos. De esta manera durante la lectura, se elimina los caracteres “DLE” añadidos durante la escritura, así, la secuencia de inicio de lista “DLE” “STX” o la secuencia final de lista “DLE” “ETX” se pueden diferenciar de estas mismas combinaciones de caracteres en los datos comprobando el número de símbolos “DLE” presentes en el código.

A continuación se explica las técnicas utilizadas en sentido inverso de la figura 22 para obtener la imagen reconstruida, es decir en el lado del decodificador.

4.2.5 Decodificador

Una vez generado el archivo comprimido es posible decodificar la imagen a diferentes calidades sacando provecho de las bondades de la codificación embebida otorgadas por el EZW. En este sentido, se puede detener el proceso de decodificación de acuerdo a un parámetro elegido por el usuario, de tal manera que con los datos extraídos hasta ese momento se obtiene una imagen a una determinada calidad.

4.2.5.1 Decodificador aritmético

Como se vio en la sección anterior, el proceso de codificación consiste en hacer más estrecho el rango de números posibles para codificar el mensaje con cada símbolo que se codifica. Ahora bien, la decodificación es el proceso inverso, es decir, consiste en expandir el rango proporcionalmente a la probabilidad de cada símbolo conforme estos se decodifican. Como se mencionó anteriormente, el primer símbolo codificado determina la

parte más significativa del número final, además, como el decodificador conoce a qué símbolo le corresponde cada rango (por el modelo de datos), sólo tiene que ver en qué rango cae el número codificado.

Volviendo al ejemplo estudiado en el codificador aritmético, el número que codifica el mensaje dado es 0.82048, debido a que este número cae dentro del rango 0.80 – 1.00, se sabe entonces que el primer símbolo del mensaje es el caracter 'T'. Ahora, puesto que se conoce los extremos (superior e inferior) del rango correspondiente al símbolo 'T', se elimina este símbolo del número codificado deshaciendo las operaciones que los introdujo.

Lo que se hace enseguida es restarle al número codificado el valor del extremo inferior del rango correspondiente al símbolo 'T', resultando el número 0.02048, luego se divide este valor por el rango del símbolo 'T', es decir 0.2, con lo cual se obtiene el nuevo valor 0.1024. Seguidamente se determina en qué subrango cae este nuevo valor, para decodificar el siguiente símbolo que es 'E'. El proceso continúa de manera similar hasta que se haya decodificado el mensaje completo.

En términos matemáticos, las relaciones utilizadas para actualizar el valor correspondiente en cada paso se expresan a continuación.

$$rango = superior(símbolo) - inferior(símbolo) \quad (4.5)$$

$$numero\_cod = numero\_cod - inferior(símbolo) \quad (4.6)$$

$$numero\_cod = numero\_cod / rango \quad (4.7)$$

Con las anteriores relaciones se obtiene el proceso completo mostrado en la tabla 4.

Tabla 4. Proceso de decodificación para el mensaje "TESIS"

Número codificado	Símbolo de salida	Límite inferior	Límite superior	Rango
0.82048	T	0.80	1.00	0.20
0.1024	E	0.00	0.20	0.20
0.512	S	0.40	0.80	0.40
0.28	I	0.20	0.40	0.20
0.4	S	0.40	0.80	0.40
0				

4.2.5.2 Decodificador EZW

El decodificador EZW permite recuperar los *coeficientes wavelet* a partir de las listas de símbolos dominantes y subordinados generadas en el módulo codificador EZW. Sus parámetros de entrada corresponden a los datos entregados por el decodificador aritmético. Es de tener en cuenta que es en esta etapa donde se selecciona qué cantidad de datos se decodifica para lograr la reconstrucción de las imágenes a distintas calidades.

4.2.5.3 IDWT o reconstrucción

En la etapa de reconstrucción se aplica la *transformada wavelet* inversa o IDWT a los coeficientes provenientes del decodificador EZW, para así obtener la imagen reconstruida, luego, a partir de esta imagen resultante puede hacerse una comparación tanto subjetiva como objetiva respecto a la imagen original, en este último caso midiendo los criterios de calidad o fidelidad tratados en la sección 3.4 del capítulo 3.

4.3 FASE DE IMPLEMENTACIÓN

La fase de implementación del algoritmo implica el desarrollo de las funciones en Matlab a través de las cuales se da funcionalidad a cada uno de los módulos diseñados en la fase previa. Algunas de estas funciones operan en conjunto con funciones proporcionadas por el *toolbox wavelet*, sin embargo, estas últimas no son descritas en detalle, pues basta con emplear el *help* de Matlab para obtener mucha más información sobre las mismas. Considerando además que la implementación total del algoritmo abarca una gran cantidad de funciones, son descritas a continuación únicamente las más importantes, sobre todo, las que implementan directamente los módulos descritos en la fase de diseño.

4.3.1 Codificador

4.3.1.1 DWT o descomposición subbanda

La aplicación de la DWT y consecuente descomposición subbanda se realiza mediante la función *dwt_imagenes* que hace uso de la función *wavedec2* del *toolbox wavelet*. Su sintaxis es *[mat,C,S] = dwt_imagenes (matriz_imagen, n, wavelet)*.

Los argumentos de entrada corresponden respectivamente a: la matriz de píxeles de la imagen original, el nivel de descomposición y el identificador de la *wavelet* respectiva. En el anexo C se encuentra una tabla en donde se especifica el identificador para las *familias wavelets* más conocidas.

Entre tanto, los parámetros de salida son: un vector de matrices *mat* correspondientes a la descomposición subbanda de la imagen, un vector *C* que contiene la estructura de *descomposición wavelet* y su correspondiente matriz de contabilidad *S*, la cual guarda información sobre las dimensiones de las subbandas.

El vector *C* del cual se hace mención, estructura los coeficientes transformados de la siguiente manera:

$$C = [\ A(n)\ |\ H(n)\ |\ V(n)\ |\ D(n)\ |\ ...\ |\ H(n-1)\ |\ V(n-1)\ |\ D(n-1)\ |\ ...\ |\ H(1)\ |\ V(1)\ |\ D(1)\ ] \qquad (4.8)$$

donde, *A* es un vector fila y representa la imagen de más baja resolución (de último nivel) que se organiza recorriendo por columnas la matriz de aproximación de *coeficientes wavelets*, análogamente *H*, *V*, y *D* son vectores fila que representan los coeficientes de detalles horizontales, verticales y diagonales respectivamente. Finalmente, la matriz *S* proporciona la siguiente información:

S(1,:): tamaño de los coeficientes de aproximación del nivel *n*.

S(i,:): tamaño de los coeficientes de detalle de nivel (n - i+2) for i = 2, ... n +1.

S(n +2, :): dimensiones de la imagen original.

4.3.1.2 Umbralización

Para el caso de umbralización global, el algoritmo evalúa el vector de descomposición *C* y encuentra entre todos los coeficientes de detalle el de mayor valor, para establecer con él un rango de selección de umbral que inicia en cero. Cuando se hace umbralización por nivel, se realiza el mismo procedimiento pero independientemente para cada una de las subbandas de detalle.

El toolbox presenta algunas estrategias predefinidas que sugieren un valor de umbral (o umbrales, en el caso de umbralización por nivel) de acuerdo a las características de la imagen y el número de coeficientes llevados a cero. La función encargada de realizar este proceso se denomina *wthrmngr*. (Para más información digitar *help wthrmngr* en el prompt de Matlab).

A continuación se describen brevemente las estrategias empleadas en el algoritmo desarrollado.

Umbralización Global

Se emplea los siguientes métodos empíricos:

- *Balance dispersión – energía*: esta opción retorna un umbral tal que los porcentajes de energía retenida y número de ceros es el mismo. La energía retenida se calcula mediante la expresión:

$$\% \text{ Energía} = 100 * (\text{norma vectorial de } CXC / \text{norma vectorial de } C)^2 \qquad (4.9)$$

Donde *C* denota la estructura de *descomposición wavelet* y *CXC* es el vector resultante luego de llevar a cero los coeficientes menores al umbral en valor absoluto. Entre tanto, la dispersión relativa en porcentaje, o número de ceros se obtiene como sigue:

% Ceros = 100(número de ceros en CXC/longitud de C)* (4.10)

- *Raíz del balance dispersión – energía*: Considera la raíz cuadrada del umbral calculado mediante el anterior método.

- *Remover cercanos a cero:* Este método establece un umbral cercano a cero calculado como el valor medio del valor absoluto de los coeficientes de detalle así: *median(abs(C))* o como *0.05*max(abs(C))* si el primer cálculo es igual a cero.

Umbralización por nivel

Las estrategias predefinidas son:

- *Método de Birgé-Massart*: tres parámetros caracterizan esta estrategia: el nivel de descomposición *J*, una constante positiva *M* y un parámetro de dispersión *a* donde $1 < a < 5$, la estrategia es tal que en el nivel *J* la aproximación es mantenida y para el nivel *j* desde 1 hasta *J*, los n_j coeficientes más grandes son conservados con la siguiente relación.

$$n_j = \frac{M}{(J + 2 - j)^a} \tag{4.8}$$

Así, la estrategia conduce a seleccionar los más altos coeficientes en valor absoluto en cada nivel, el número de coeficientes conservados crecen escasamente con *J - j*. De acuerdo a los estudios de Birgé y Massart, un valor típico para compresión es $a = 1.5$. Sea *D* el tamaño (dimensiones) de los coeficientes de aproximación, se tiene tres opciones que dependen del valor de *M* de la siguiente manera: para escasez alta *M = 4*prod(D)*, para escasez media *M = 4*4*prod(D) / 3*, y para escasez baja *M = 4*8*prod(D) / 3*.

- Métodos empíricos

A través de las mismas estrategias aplicadas en la umbralización global, se calcula los umbrales para cada subbanda y nivel de descomposición.

4.3.1.3 Codificador EZW

El algoritmo EZW se implementa mediante las funciones descritas a continuación.

Función *cod_ezw*

Esta función realiza la codificación EZW de la imagen transformada. Los argumentos de entrada son la matriz de *coeficientes wavelet* y un entero *N* que indica el número de niveles EZW descartados en la codificación. El primer paso es calcular el umbral inicial según la ecuación (4.1), es de tener en cuenta que éste umbral es requerido para sincronizar el codificador y el decodificador, y por tanto, se debe transmitir su valor en el archivo que almacena los datos de la imagen comprimida, luego se calcula el orden de extracción de los *coeficientes wavelet* mediante la función *oe_Morton*.

Función *oe_Morton*

Esta función se encarga de calcular el orden de extracción de los *coeficientes wavelet* de la imagen transformada. El orden de extracción implementado es el denominado *orden de extracción de Morton*, ahora, como argumento de entrada la función requiere el tamaño de la imagen transformada y como parámetro de salida retorna una matriz *A*, donde cada elemento *A(i,j)* indica el orden de extracción para el *coeficiente wavelet* correspondiente (con igual posición en la imagen transformada) en la imagen transformada.

La pasada dominante y pasada subordinada se implementan mediante las funciones *pasada_D* y *pasada_S* respectivamente.

Función *pasada_D*

Esta función es invocada una vez por cada umbral de análisis EZW y se encarga de codificar los *coeficientes wavelet* que aún no hayan sido codificados en pasos anteriores. Es aquí donde se generan los símbolos 'p' y 'n' para los coeficientes significativos respecto

al umbral actual y cuyas magnitudes se almacenan en la denominada *lista subordinada*; o los símbolos 'z' o 't' en caso contrario. Cuando se da ésta última situación, primero se comprueba si el coeficiente pertenece a un árbol de ceros; si es así, no se añade ningún símbolo a la lista de *símbolos dominantes* para ese coeficiente. En caso de no pertenecer a un árbol de ceros, se verifica si se trata de un cero aislado, o de una raíz de un árbol de ceros; ello se hace mediante las funciones *descendencia* y *arbol_ceros.*

Función *descendencia*

Mediante la función *descendencia* se comprueba si un coeficiente dado es raíz de un árbol de ceros, comparando el valor de sus coeficientes descendientes con respecto al umbral actual. En caso de que alguno de los descendientes sea significativo, la función retorna el valor 1, indicando a la función *pasada_D* que el coeficiente no es una raíz de un árbol de ceros para que sea codificado como 'z'. En caso de que todos sus descendientes sean insignificantes respecto al umbral actual, retorna un 0, y el coeficiente se codifica como una raíz de un árbol de ceros, adicionalmente, para este último caso, se identifica a todos los descendientes del coeficiente como miembros de un árbol de ceros escribiendo un 1 en las posiciones respectivas dentro de una matriz de referencia *R*, creada especialmente para éste propósito; la función *arbol_ceros* cumple con esta funcionalidad.

Finalmente, el resultado de la función *pasada_D* consiste en una lista de símbolos *dominantes* ('z', 't', 'n' o 'p'), correspondiente a la codificación EZW de los *coeficientes wavelet* para el umbral (o nivel EZW) de codificación actual.

Función *pasada_S*

Esta función implementa la pasada de refinamiento, que consiste en extraer el siguiente bit más significativo de cada uno de los coeficientes de la lista subordinada, es decir, de los coeficientes identificados como significativos en pasadas dominantes anteriores. Durante esta pasada, se compara la magnitud de cada coeficiente con el umbral actual, y se codifica un ‘1’ si ésta es mayor o igual que el umbral, o un ‘0’ en caso contrario. La salida de la

función *pasada_S* es una lista de símbolos denominados *símbolos subordinados* (‘1’ o ‘0’), correspondientes a la codificación subordinada de los *coeficientes wavelet* para el umbral (o nivel EZW) de codificación actual.

Luego de la pasada subordinada, la función *cod_ezw* reduce el umbral a la mitad y realiza nuevamente el procedimiento anterior. El proceso de codificación EZW termina cuando el umbral coincide con el parámetro 2^N.

4.3.1.4 Codificador aritmético

A continuación se explica en forma detallada el modo de operación de las funciones que se implementan para la codificación aritmética.

Función *cod_arit*

La función *cod_arit* es la encargada de realizar la codificación aritmética. Tiene un único argumento de entrada que es la cadena de símbolos entregados por la etapa anterior de la codificación EZW, tanto símbolos dominantes como subordinados. Al final esta función entrega una cadena codificada aritméticamente y una tabla de estadísticas al finalizar la correspondiente etapa de codificación.

Debido a que el codificador aritmético recibe cadenas de símbolos 'z', 't', 'n' o 'p', el primer paso que realiza es asignar un número entero a cada uno de estos símbolos de entrada, esta operación se realiza mediante la función *caracter_a_entero*.

Función *caracter_a_entero*

Esta función asigna un número entero a cada símbolo del alfabeto de entrada de la siguiente forma: al símbolo 'z' le asigna el número 0, al símbolo 't' el número 1, al símbolo 'n' el 2 y finalmente al símbolo 'p' le corresponde el número entero 3. Como argumento de entrada

esta función tiene un caracter que es el símbolo original del mensaje, y como parámetro de salida tiene el número entero asignado al símbolo en cuestión. Este valor entero sirve de índice para buscar el rango asignado al símbolo en el arreglo modelo de datos mencionado anteriormente (*modelo*). En caso que un símbolo de entrada no sea ninguno de los cuatro posibles EZW, se produce un error.

El paso a seguir es obtener las estadísticas del respectivo símbolo de entrada según el índice correspondiente de la función anterior, lo cual se realiza mediante la función *entero_a_simbolo*.

Función *entero_a_simbolo*

Como ya se mencionó esta función es la encargada de obtener las estadísticas del símbolo de entrada, es decir, con ella se obtiene la escala, el valor superior y el valor inferior de dicho símbolo. Para su implementación se necesita un único argumento de entrada que es el número entero correspondiente al símbolo original del mensaje, y como parámetro de salida se tiene una estructura que contiene los tres datos mencionados previamente, dichas estadísticas se obtienen leyendo la posición del símbolo en el arreglo (*modelo*) de estadísticas.

Función *codificar_simbolo*

Ya en este punto del proceso se realiza ahora sí la codificación aritmética como tal de cada símbolo (recordar que en la codificación aritmética los símbolos no se codifican uno a uno, cuando se habla de codificación aritmética de cada símbolo se refiere mas bien al procesamiento de cada uno de ellos). Pues bien, para tal efecto se implementa esta función, la cual actualiza los valores de "*superior"* y "*inferior"* del rango dentro del cual estará el número codificado de acuerdo con las probabilidades del símbolo actual, esto lo hace mediante las relaciones (4.1) a (4.3). Es importante mencionar que para realizar la codificación en este módulo se emplea aritmética entera sin signo de 16 bits, ello para tener el número de salida con tantos bits como sean necesarios para codificar el mensaje, es por

ello que esta función también se encarga de evaluar y evitar un posible error de desbordamiento (*underflow*) el cual es muy típico de darse cuando se trabaja con este tipo de aritmética [35, 36].

Los argumentos de entrada de esta función son los valores "*superior"*, "*inferior*" y "*escala*" del símbolo actual. Al final, como parámetro de salida se tiene una cadena donde se almacena el símbolo respectivo codificado aritméticamente.

El paso a seguir es actualizar el modelo de datos tras la codificación de cada símbolo para poder predecir la probabilidad de los símbolos de una manera precisa y realizar así una óptima compresión de datos, cumpliendo de esta manera con las premisas de un modelo de datos eficiente mencionadas la sección 4.2.4.4. Se describe a continuación la función que realiza dicho proceso.

Función *actualizar_modelo*

Esta función actualiza la tabla de probabilidades tras la codificación de cada símbolo de forma que represente de manera fiel las probabilidades de ocurrencia de cada símbolo en el mensaje de entrada. El argumento de entrada de esta función es el símbolo codificado y no tiene parámetros de salida, es decir, solo es necesario invocar esta función para que realice el objetivo planteado.

Finalmente, una vez se haya codificado todo el mensaje, es necesario adicionar a la cadena de salida los bits que queden en el codificador aritmético. Con este propósito se implementa la función *clarear_codificador*, la cual devuelve el mensaje final codificado aritméticamente.

4.3.1.5 Datos binarios

Para el desarrollo algorítmico del módulo de datos binarios, se implementan las funciones que se explican a continuación.

Función *escribir_archivo*

Esta función es la encargada de escribir el archivo comprimido de la imagen original, es decir, escribe los datos de la imagen comprimida en un archivo. Los argumentos de entrada para esta función son tres: el nombre de la imagen original, la ruta donde se guarda el archivo comprimido, y la estructura que contiene la imagen comprimida.

Para que haya una correspondencia al orden del proceso especificado en la sección 4.2.4.5, el primer paso que se realiza es escribir la cabecera del archivo, esto se hace simplemente leyendo los diferentes campos de la estructura que contiene la imagen comprimida y que van dentro del cuerpo de la cabecera, es decir: tamaño de la imagen original, los datos de la matriz *S*, el tipo de *transformada wavelet*, y el número de niveles de *descomposición wavelet*.

A continuación de la información correspondiente a la cabecera, se escribe el umbral inicial EZW, para finalmente proceder con la escritura de las listas de datos tanto dominantes como subordinadas ordenadas por niveles EZW, (ver figura 25), es importante mencionar que en la secuencia del algoritmo, primero se opera con la lista dominante perteneciente al nivel EZW respectivo.

Para llevar a cabo lo anterior, se debe en primer lugar convertir las cadenas de bits a caracteres para escribir las listas como *char*, en este punto hay que recordar que ambos tipos de listas (dominantes y subordinadas) son cadenas de bits de longitud variable, pues han sido codificadas aritméticamente de la misma forma. Ese procedimiento de conversión de bits a caracteres para escribir listas como *char* es realizado por la función *bits_a_caracteres*.

Función *bits_a_caracteres*

Como se mencionó, esta función es la encargada de convertir una cadena binaria en una cadena de caracteres para su escritura en el archivo, la cual tiene como argumento de entrada la cadena binaria de entrada. Ahora bien, según la teoría desarrollada en la sección 4.2.4.5, para escribir estas cadenas de bits en dicho archivo, se agrupan los bits en grupos de ocho (8), es por esta razón que en primer lugar la función en estudio comprueba si el número total de bits en la lista respectiva es múltiplo de ocho, de no ser así, rellena con ceros para obtener al final un número entero de grupos de bits con el tamaño indicado.

Una vez se han agrupado los datos como se acaba de explicar, se leen los bits de la lista de entrada de 8 en 8 y se guardan en una variable para después escribir el símbolo ASCII correspondiente a ese entero en la cadena de salida.

Función *escribir_lista*

Finalmente para completar el proceso, se necesita una forma de indicar al decodificador dónde empieza y termina cada lista EZW utilizando como se mencionó la técnica de "*caracteres de relleno*", pues bien, ésta es la función implementada para tal fin, la cual coloca al inicio de cada lista los caracteres ASCII "DLE" y "STX" y al final de la misma los caracteres "DLE" y "ETX".
Por último dentro del cuerpo de la función *escribir_archivo* se realiza un procedimiento análogo al explicado previamente para las listas subordinadas.

Ahora, para la decodificación del archivo comprimido obtenido mediante la secuencia que se acaba de desarrollar, se implementan las siguientes funciones:

Función *leer_archivo*

Esta función lee los datos almacenados en el archivo comprimido de la imagen para su decodificación, además brinda la opción de seleccionar el número de niveles de

decodificación EZW empleados, de manera que se puede decodificar el mismo archivo a diferentes calidades y tasas binarias.

Los argumentos de entrada de esta función son: el nombre del archivo comprimido y el número de niveles EZW descartados durante la decodificación, y como parámetro de salida tiene la estructura que contiene la imagen comprimida.

Básicamente el proceso realizado por esta función es "similar" que el hecho por la función *escribir_archivo*, con la diferencia que esta última escribía los datos de sincronismo, los umbrales EZW y las listas EZW en un archivo, mientras que la función *leer_archivo*, hace una lectura de esos mismos parámetros respecto del mismo archivo. Tanto la información de sincronismo como los umbrales, son leídos directamente, pero para las listas EZW, se debe hacer una conversión de caracteres a bits, lo cual se implementa con la función siguiente.

Función *caracteres_a_bits*

Esta función convierte una cadena de caracteres en una cadena binaria, además desagrupa los bits que forman cada uno de los símbolos ASCII de la cadena de caracteres de entrada. Es decir, esta es la función inversa a la *bits_a_caracteres*, y devuelve como se mencionó una cadena binaria.

Función *leer_lista*

Finalmente para completar el proceso de decodificación del archivo se implementa esta función cuyo objetivo es leer una lista dominante o subordinada desde el archivo de imagen comprimido, además elimina los caracteres de relleno empleados durante la escritura de las listas. Tiene como argumento de entrada un identificador del archivo comprimido y sus parámetros de salida son la lista leída y un entero para indicar que se ha leído todo el archivo. Para obtener este último parámetro de salida se examina los caracteres ASCII "DLE", "STX" y "ETX" colocados por la función *escribir_lista*, comprobando

adicionalmente que los caracteres “DLE” no sean accidentales y que puedan originar una mala interpretación del comienzo y/o fin de una secuencia dada, tal como se explicó en la sección 4.2.4.5.

4.3.2 Decodificador

Para cumplir con el propósito de la decodificación de la imagen a diferentes calidades, se emplea la función *decodificador*, cuyos argumentos de entrada son el nombre del archivo comprimido y el número de niveles EZW descartados durante la decodificación. Esta función es la encargada, luego de extraer los datos del archivo comprimido, de invocar en primera instancia a la función *decod_arit* que realiza la decodificación del mensaje codificado aritméticamente, para luego obtener los datos decodificados EZW empleando la función *decod_ezw*. Como parámetro de salida la función *decodificador* retorna una matriz de *coeficientes wavelet* que es procesada por el algoritmo para finalmente visualizar la imagen reconstruida.

4.3.2.1 Decodificador aritmético

Ya se estudió el modo de operación de las funciones que se implementan para la codificación aritmética. A continuación se explica del mismo modo las funciones implementadas en el módulo de decodificación aritmética.
Función *decod_arit*

La función *decod_arit* realiza la decodificación aritmética de un mensaje, la cual a partir del número codificado, devuelve el mensaje original, el argumento de entrada es el mensaje codificado y como parámetro de salida tiene el mensaje decodificado. El primer paso que se realiza en esta función es inicializar (igualar a cero) los valores de la estructura que contiene la escala, el valor superior e inferior del símbolo que está siendo decodificado.

La secuencia y el proceso de decodificación consisten simplemente en reproducir algorítmicamente el análisis descrito en palabras en la sección 4.2.5.1. Es decir, mientras que la codificación consistía en hacer más estrecho el rango de números posibles para codificar el mensaje con cada símbolo que se codificaba, la decodificación consiste en expandir el rango proporcionalmente a la probabilidad de cada símbolo conforme estos se leen. Esta tarea es realizada mediante las siguientes funciones.

Función *decodificar_simbolo*

La función *decodificar_simbolo* es la encargada de devolver y actualizar el valor del número codificado de acuerdo al valor del límite superior, inferior y el rango del símbolo de salida respectivo, para este propósito la función implementa las relaciones (4.4) a (4.6), en este punto es importante recordar nuevamente que todo el proceso se realiza en aritmética entera sin signo de 16 bits. El número codificado del cual se hace mención es el que determina el rango correspondiente a un símbolo específico y por ende se utiliza para decodificar el símbolo actual según el valor del arreglo (modelo de datos) *modelo*.

Función *simbolo_a_entero*

Esta función devuelve el número entero correspondiente al símbolo original codificado a partir del valor hallado en la función anterior, para ello, recorre la tabla de estadísticas *modelo*, para encontrar la posición del símbolo original. Como se mencionó, el parámetro de salida de esta función es el número entero correspondiente al símbolo original. El proceso continúa deshaciendo los cambios introducidos en el número codificado al codificar el símbolo actual, esto es, actualizar el rango, extraer los bits que coincidan en "superior" e "*inferior*", y actualizar el valor de "*cod*" a partir de la cadena de bits de entrada. Con este propósito se emplea la función *actualizar_cadena*.

Función *actualizar_cadena*

En esta parte del algoritmo se actualiza el valor de la cadena de entrada tras cada símbolo decodificado, de manera que el valor del número codificado pertenezca al rango del siguiente símbolo a decodificar. El argumento de entrada en esta función es el mensaje codificado aritméticamente, y el parámetro de salida es el mensaje codificado aritméticamente tras la eliminación del último símbolo.

Función *entero_a_caracter*

Esta es la función que realiza el proceso inverso a la función *caracter_a_entero* de la etapa de codificación, es decir, devuelve el símbolo EZW correspondiente al número entero decodificado (al número 0 le asigna el símbolo 'z' y así sucesivamente). Como argumento de entrada tiene el número entero asignado al símbolo original y como se dijo, retorna el símbolo original del mensaje.

Finalmente, el último paso consiste en actualizar el modelo de datos tras la decodificación de cada símbolo. La función encargada de dicho proceso es la misma que se describió en la etapa de actualización de modelo del módulo de codificación y que corresponde a *actualizar_modelo*.

4.3.2.2 Decodificador EZW

La función *decod_ezw* realiza la decodificación EZW de la matriz de *coeficientes wavelet*, a partir de las listas de símbolos dominantes y subordinados para cada nivel de codificación EZW. Sus argumentos de entrada son una estructura que contiene las listas de símbolos dominantes y subordinados para cada nivel de codificación EZW resultantes de la codificación, y el tamaño de la imagen.

El proceso de decodificación EZW empieza inicializando una matriz de coeficientes decodificados a cero. Para cada nivel de decodificación, se realiza una pasada dominante inversa y una pasada subordinada inversa (excepto para el último nivel, en el que sólo se realiza la pasada dominante). Durante la pasada dominante inversa, cada vez que se encuentra un símbolo significativo (‘p’ o ‘n’), se le asigna un valor al coeficiente en cuya posición aparezca dicho símbolo, que está relacionado con el umbral actual, ahora, durante la pasada subordinada inversa, se actualiza el valor de los coeficientes decodificados, según el símbolo subordinado correspondiente, de manera que se reduce a la mitad el intervalo de incertidumbre al que pertenece el coeficiente. Las funciones encargadas de este proceso son *pasada_D_inversa* y *pasada_S_inversa*.

Función *pasada_D_inversa*

Para cada nivel de decodificación EZW, se realiza una pasada dominante inversa, mediante la cual se obtiene un valor aproximado de los coeficientes codificados durante la pasada dominante.

Cuando el símbolo dominante actual es ‘p’, el coeficiente transformado correspondiente es un valor positivo, perteneciente al intervalo $[T_0, 2T_0)$, donde T_0 es el umbral actual, por tanto, se le asigna el valor $1.5T_0$, que coincide con la mitad del intervalo de incertidumbre, minimizando de esta manera el error cuadrático medio (MSE). Cuando el símbolo dominante actual es ‘n’, el coeficiente transformado correspondiente es un valor negativo, perteneciente al intervalo $(-2T_0, -T_0]$, y se le asigna el valor $-1.5T_0$. Si el símbolo actual es una ‘t’, significa que el coeficiente es una raíz de un árbol de ceros, y por tanto se marca con un ‘1’ la posición de sus hijos en la matriz de referencia *R*, finalmente si el coeficiente es una ‘z’, el coeficiente es un cero aislado. Tras la decodificación del coeficiente, se marca con un ‘1’ su posición en la matriz de referencia, para indicar que el coeficiente ya ha sido decodificado.

Función *pasada_S_inversa*

Ésta función actualiza el valor de los coeficientes decodificados. Como se vio en la función *pasada_D_inversa*, durante la pasada dominante inversa se asigna a cada coeficiente el valor medio del intervalo $[T_0, 2T_0)$, si el coeficiente es positivo, o del intervalo $(-2T_0, -T_0]$, en caso de que el coeficiente sea negativo. Mediante las sucesivas pasadas subordinadas inversas, se actualiza el valor de cada coeficiente, de manera que se reduce el intervalo de incertidumbre a la mitad, y por tanto, el error cuadrático medio (MSE).

En caso de que el coeficiente sea positivo y el símbolo subordinado actual para ese coeficiente sea '1', significa que el valor del coeficiente pertenece al subintervalo superior $[3T_0/2, 2T_0)$, mientras que si el símbolo subordinado es '0', el valor del coeficiente pertenece al subintervalo inferior, $[T_0, 3T_0/2)$, ahora bien, en caso de que el coeficiente sea negativo y el símbolo subordinado actual para ese coeficiente sea '1', significa que el valor del coeficiente pertenece al subintervalo $(-2T_0, -3T_0/2]$, mientras que si el símbolo subordinado es '0', el valor del coeficiente pertenece al subintervalo $[-T_0, -3T_0/2)$.

4.3.2.3 IDWT o reconstrucción

Así como para la *descomposición wavelet* se implementa la función *dwt_imagenes* que hace uso de la función *wavedec2* del toolbox, para la reconstrucción se implementa la función *idwt* que emplea la función *waverec2,* siendo esta última la encargada de realizar el proceso inverso al de *wavedec2*, es decir, generar una imagen reconstruida a partir de los *coeficientes wavelets*. Lógicamente, en este punto del sistema y debido a las etapas previas de umbralización y cuantificación / decuantificación, los *coeficientes wavelet* entregados como argumento de entrada, ya no son los mismos inicialmente calculados por *wavedec2*.

La sintaxis empleada en el algoritmo es *x = waverec2(CXC,S,wavelet)*, que indica la reconstrucción de la matriz *x* a partir de la estructura de *descomposición wavelet* procesada *[CXC,S]* (inicialmente *[C,S]*).

La medida de los criterios de fidelidad se realiza invocando la función *c_fidelidad* cuya notación es *[MSE, Erms, SNR, PSNR] = c_fidelidad (img_orig, img_recons)*, los argumentos de entrada son la imagen original y la imagen reconstruida respectivamente, mientras que los parámetros de salida son: el error cuadrático medio (*MSE*), el error cuadrático medio *rms* (*Erms*), la relación seña a ruido (*SNR*) y la relación señal a ruido pico (*PSNR*).

4.4 CARACTERÍSTICAS

Las características de implementación del algoritmo desarrollado pueden resumirse en los siguientes puntos.

- Es un algoritmo de compresión de imágenes fijas en escala de grises que aplica un esquema *lossy* mediante codificación por *transformada wavelet*.

- Implementa la DWT como método de compresión de la energía de la imagen y aprovecha sus ventajas traducidas en la compactación de energía y la generación de un gran porcentaje de *coeficientes wavelet* próximos a cero.

- Emplea como método de cuantificación el algoritmo EZW el cual aprovecha la naturaleza jerárquica de la *descomposición wavelet* y hace posible la codificación embebida.

- Implementa como método de codificación de entropía un codificador aritmético que es con el que mejores factores de compresión se obtiene, además de ofrecer un buen desempeño cuando se combina con un modelo de datos adaptativo.

Entre sus principales características funcionales se encuentran:

- Permite visualizar la descomposición subbanda como consecuencia de la aplicación de la DWT sobre una imagen empleando diferentes *familias wavelets*, lo cual hace posible su estudio comparativo.

- De acuerdo a la descomposición subbanda, hace posible la umbralización de la imagen transformada (en el plano tiempo - escala) mediante un umbral global o diferentes umbrales que el usuario puede definir de acuerdo a la calidad de imagen o tasa de compresión deseada.

- Presenta medidas objetivas de la calidad de la imagen reconstruida respecto a la imagen original para lo cual implementa los denominados criterios de fidelidad tales como el error cuadrático medio y la relación señal a ruido.

- Como consecuencia de la codificación embebida permite la decodificación a diferentes calidades de imagen a partir de un mismo archivo comprimido.

4.5 MODO DE EJECUCIÓN DEL ALGORITMO

Para la ejecución del algoritmo implementado se tiene dos posibilidades. La primera de ellas es acceder a la GUI ejecutando el comando *guide* en el prompt, lo que despliega una interfaz propia de Matlab que permite abrir una GUI existente seleccionando la opción 'Open Existing GUI'. Mediante el browser se selecciona el archivo *algoritmo_wavelets.fig* y se pone en funcionamiento dando clic en el botón '*Run*' ubicado en la parte superior del *Layout Editor*.

La segunda posibilidad consiste simplemente en ejecutar en el prompt la función cuyo nombre es el mismo con el que se identifica al archivo *.fig* generado en el proceso de

desarrollo mediante GUIDE, es decir, basta con ejecutar el comando *algoritmo_wavelets* para inicializar al algoritmo, siempre y cuando el *path* de trabajo de Matlab esté direccionado al directorio de la aplicación. En el anexo D se presenta a manera de manual de usuario detalles del empleo del algoritmo.

4.6 LIMITACIONES

El mayor inconveniente del algoritmo implementado radica en el tiempo de procesamiento del algoritmo EZW, debido al concepto de aproximaciones sucesivas en el que se fundamenta y que se implementa mediante las pasadas dominantes y sucesivas para cada uno de los umbrales que decrecen a la mitad hasta llegar a un valor de uno. Como es de esperarse, la pasada dominante implica un mayor procesamiento, pues es allí donde se realiza la asignación de símbolos a cada uno de los *coeficientes wavelet* luego de hacer a cada uno de ellos el análisis explicado previamente. Para alivianar un poco este problema, el algoritmo presenta la posibilidad de fragmentar la imagen transformada (imagen de *coeficientes wavelet*) en bloques y procesarlos a partir de la codificación EZW en adelante independientemente, lo cual reduce de manera significativa el tiempo de procesamiento.

Si bien el algoritmo permite trabajar con cualesquiera de los formatos de imágenes soportados por Matlab, presenta una limitación referente al tipo de imágenes, pues éste ha sido diseñado para procesar solo imágenes indexadas y en escala de grises, de tal manera que no permite procesar imágenes tipo *truecolor*.

En cuanto al proceso de análisis que permite visualizar la descomposición subbanda de la imagen se tiene la limitación para imágenes de tamaño mayor a 32x32 píxeles de poder observar tal descomposición a un nivel mayor a cinco (5) puesto que por motivos de representación gráfica se ha limitado hasta este valor; sin embargo, ello no supone un grave inconveniente puesto que para este nivel ya se cuenta con un total de quince subbandas de detalle propicias para la experimentación.

5 SIMULACIÓN, PRUEBAS Y RESULTADOS

En este capítulo se presenta un informe de las pruebas realizadas con imágenes fijas y los resultados obtenidos indicando las medidas de criterios objetivos, además, en algunos casos, se anexan las imágenes reconstruidas para apreciar subjetivamente si el nivel de degradación visual es o no considerable.

Para estudiar el rendimiento de la aplicación, se han realizado pruebas de compresión con distintas imágenes a diferentes parámetros de compresión, esencialmente, empleando distintas *wavelets*, distintos niveles de descomposición subbanda, y umbralización de tipo global. Cabe aclarar que no se ha considerado mostrar los resultados para la umbralización por nivel debido a que para este caso resultaría un gran número de pruebas dependiendo del nivel de descomposición aplicado, así por ejemplo para el nivel cinco se tendría que fijar adecuadamente tres umbrales por cada nivel, conllevando esto a la generación de mucha información, no obstante, es de resaltar que este tipo de umbralización es importante en aplicaciones donde se requiera preservar determinados tipos de detalle.

5.1 IMÁGENES DE DIMENSIONES MEDIAS

Las imágenes seleccionadas para realizar la experimentación inicial se denominan *casa* y *torre*, ambas imágenes de tipo escala de grises, de 512 x 512 píxeles y en formato TIF, formato elegido porque las imágenes con esta extensión no presentan compresión.

Tal como puede apreciarse en la figura 26, la imagen *casa.tif* presenta detalles marcados, mientras que la imagen *torre.tif* se caracteriza por presentar áreas más uniformes.

a **b**

Figura 26. Imágenes de tamaño 512 X 512 empleadas en la experimentación: a) *casa.tif* b) *torre.tif*

5.1.1 Nivel de descomposición bajo

Partiendo de la consideración de dos imágenes visiblemente distintas, se plantea un primer caso de análisis correspondiente al empleo de un solo nivel de descomposición de las imágenes, es decir, la generación de las cuatro subbandas, LL_1, LH_1, HH_1 y HL_1, de las cuales, la primera se conserva a través de cada módulo del algoritmo, mientras que las otras tres pueden ser alteradas mediante el proceso de umbralización. A continuación se presenta los resultados obtenidos a este nivel de descomposición empleando diferentes *wavelets*, las cuales se han seleccionado experimentalmente entre las que concentran mayor cantidad de energía en la subbanda de aproximación (LL_n), sin embargo se presenta solo las imágenes reconstruidas con la *wavelet Haar*, ello con el propósito de establecer un punto de comparación para una misma *wavelet* entre el presente nivel de descomposición y uno mayor, analizado en la siguiente sección.

Imagen: *casa.tif*

Peso en Bytes: 262504

Nivel de descomposición: 1

Tabla 5. Análisis de *casa.tif* para nivel 1 y estrategia *Remover cercanos a Cero*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	3.50	99.99	36.19	47.28	1.22	1.33:1
Daubechies 5	3.50	99.99	40.98	47.03	1.29	1.38:1
Symlet 2	3.50	99.99	39.53	47.14	1.26	1.38:1
Coiflet 3	3.50	99.99	41.57	46.96	1.31	1.35:1

De acuerdo a la tabla 5, es evidente que por tratarse de una *descomposición wavelet* de primer nivel y por emplear la estrategia que sugiere un umbral próximo a cero, los valores MSE son pequeños y además no se logra factores de compresión altos. De este modo, las imágenes reconstruidas pueden considerarse de alta calidad, así por ejemplo, bajo las medidas de la tabla anterior para la *wavelet Haar,* la imagen reconstruida se visualiza como lo presenta la figura 27.

Imagen original

Imagen Reconstruida

Figura 27. Imagen original y reconstruida de *casa.tif* para nivel 1 y estrategia *Remover cercanos a cero*

Empleando ahora la estrategia *Balance dispersión – energía* que establece mayores umbrales buscando que el porcentaje de energía conservada y el porcentaje de ceros (dispersión) coincidan, se obtiene los datos consignados en tabla 6.

Tabla 6. Análisis de *casa.tif* para nivel 1 y estrategia *Balance dispersión - energía*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	162.50	99.43	75.00	28.88	84.16	3.23:1
Daubechies 5	163.52	99.66	75.00	31.02	51.40	3.05:1
Symlet 2	143.54	99.59	75.00	30.26	61.23	3.13:1
Coiflet 3	146.65	99.67	75.00	31.14	50.05	2.97:1

La información proporcionada por esta tabla, revela que no se logra un balance entre la dispersión y el porcentaje de energía, lo cual significa que en la *descomposición wavelet* de primer nivel de la imagen, el aporte de energía de los coeficientes de detalle a la imagen transformada no es tan significativo comparativamente con el aporte de energía de los coeficientes de aproximación. Además, los datos de la tabla 6 respecto al porcentaje de ceros indican que para este caso se elimina la totalidad de coeficientes de detalle, los cuales para el nivel de descomposición en consideración representan las tres cuartas partes de la imagen transformada (75 %).

Analizando las tablas 5 y 6, para una *wavelet* en particular se aprecia que cuando se aplica un umbral mayor, como en el caso de la estrategia *Balance dispersión – energía* respecto a la estrategia *Remover cercanos a cero*, el porcentaje de ceros se incrementa, resultado esperado puesto que dichos ceros en su gran mayoría representan los coeficientes que en el proceso de umbralización son fijados a cero (algunos coeficientes adquieren valor cero por el efecto mismo de la transformada), o en otros términos, los coeficientes menores en valor absoluto al umbral de análisis.

La figura 28 presenta la imagen original y la imagen reconstruida con la *wavelet Haar* siendo éste el peor caso de los resultados de la tabla 6 (las imágenes reconstruidas para los otros resultados de la tabla, suponen una mejor calidad subjetiva).

Imagen original

Imagen Reconstruida

Figura 28. Imagen original y reconstruida de *casa.tif* para nivel 1 y estrategia *Balance dispersión - energía*

Comparando los resultados de las tablas 5 y 6, se confirma que empleando un valor de umbral pequeño, se obtiene buena calidad de imagen, pero a cambio de ello, un factor de compresión bajo, mientras que, empleando un valor de umbral grande, se logra un factor de compresión mayor a cambio de una imagen reconstruida de menor calidad. No obstante, la imagen reconstruida de la figura 28 puede considerarse visualmente de buena calidad, lo cual se justifica teniendo en mente que la *descomposición wavelet* es de primer nivel, y por ende, los coeficientes de aproximación (siempre preservados) contienen la mayor parte de la energía de la imagen transformada. Generalizando los resultados, los factores de compresión no son altos, sin embargo y como justificación a ello, la diferencia entre la imagen original y la imagen reconstruida resulta difícilmente apreciable por el ojo humano.

Las mismas pruebas realizadas para el nivel 1 de *descomposición wavelet* a *casa.tif* se han desarrollado para la imagen *torre.tif*. La tabla 7 contiene la información recolectada empleando la estrategia *Remover cercanos a cero*.

Imagen: *torre.tif*

Peso en Bytes: 262506

Nivel de descomposición: 1

Tabla 7 Análisis de *torre.tif* para nivel 1 y estrategia *Remover cercanos a cero*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	1.0	100	20.22	57.17	0.12	1.44:1
Daubechies 5	1.0	100	54.90	53.03	0.32	1.92:1
Symlet 2	1.0	100	54.60	53.01	0.33	1.96:1
Coiflet 3	1.0	100	55.19	53.01	0.33	1.87:1

Tal como ya se expresó, la imagen *torre.tif* presenta algunas áreas suaves (caracterizadas por no tener demasiada variación de los valores entre píxeles vecinos), lo cual la convierte en una imagen propicia para un proceso de compresión puesto que en una imagen con estas características se presenta mayor redundancia entre píxeles. Los resultados de la tabla 7 reflejan esta situación en el sentido que realizando una comparación *wavelet* a *wavelet* con la tabla 5 se obtiene factores de compresión un poco más altos que los alcanzados para la imagen *casa.tif*. La figura 29 muestra la imagen original y la reconstruida caracterizada de acuerdo a tabla 7 empleando la *wavelet Haar*.

Imagen original Imagen Reconstruida

Figura 29. Imagen original y reconstruida de *torre.tif* para nivel 1 y estrategia *Remover cercanos a cero*

De igual manera que para la otra imagen, el error cuadrático medio es mínimo y la calidad de la imagen reconstruida es bastante buena subjetivamente, calidad que puede ser disminuida un poco en aras de lograr mejores factores de compresión. En este orden de ideas, se emplea la estrategia *Balance dispersión – energía* obteniendo los resultados de la tabla 8.

Tabla 8. Análisis de *torre.tif* para nivel 1 y estrategia *Balance dispersión – energía*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	185.50	99.74	75	32.33	38.02	3.77:1
Daubechies 5	193.01	99.86	75	34.75	21.77	3.52:1
Symlet 2	182.83	99.83	75	33.98	26.02	3.64:1
Coiflet 3	169.12	99.87	75	34.88	21.15	3.42:1

La imagen reconstruida con las características de la tabla anterior para la *wavelet Haar* es la que se muestra en la figura 30.

Imagen original Imagen Reconstruida

Figura 30. Imagen original y reconstruida de *torre.tif* para nivel 1 y estrategia *Balance dispersión – energía*

Una vez más puede concluirse que el efecto de aplicar un umbral mucho mayor no resulta crítico en cuanto a la degradación visual de la imagen reconstruida por el hecho de descomponer la imagen original un solo nivel. En la siguiente sección se analiza el efecto de realizar la descomposición iterativa sobre la subbanda de aproximación resultante en cada paso de descomposición.

5.1.2 Nivel de descomposición alto

Un nuevo caso de experimentación se plantea ahora para la *descomposición wavelet* (o subbanda) a un nivel superior, esto con el propósito de analizar los resultados comparativamente con los obtenidos en la compresión de imágenes aplicando el primer nivel de descomposición. El nivel de descomposición aplicado es cinco (valor máximo permitido por el algoritmo) y las pruebas realizadas son similares a las de la sección anterior.

En primera instancia, aplicando la estrategia que plantea un valor de umbral cercano a cero, cuya información se muestra en la tabla 9 para la imagen *casa.tif*, se aprecia que los resultados son semejantes a los presentados para la descomposición *wavelet* de primer

nivel, y de igual manera, las imágenes reconstruidas empleando diferentes *wavelets* pueden catalogarse como de buena calidad. Sin embargo, para este caso de análisis la descomposición *wavelet* (o subbanda) es de nivel cinco, lo cual implica que un total de quince subbandas de detalle son generadas, representando en conjunto un porcentaje de energía mayor que el aportado por las únicas tres subbandas de detalle correspondientes a la descomposición *wavelet* de primer nivel.

Imagen: *casa.tif*
Peso en Bytes: 262504
Nivel de descomposición: 5

Tabla 9. Análisis de *casa.tif* para nivel 5 y estrategia *Remover cercanos a cero*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	3.50	99.99	41.66	46.71	1.39	1.41:1
Daubechies 5	3.50	99.99	45.26	46.47	1.47	1.38:1
Symlet 2	3.50	99.99	44.87	46.54	1.44	1.45:1
Coiflet 3	3.50	100	44.95	46.40	1.49	1.28:1

Empleando ahora la estrategia *Balance dispersión – energía* se obtiene la información de la tabla 10, la cual refleja que aplicando umbrales elevados, se logra factores de compresión mayores, pero el error cuadrático medio (MSE) se torna relativamente grande. Como consecuencia entonces la degradación visual de la imagen ahora se hace muy notable, lo cual puede evidenciarse en la figura 31 que ilustra la imagen reconstruida empleando la *wavelet Haar*.

Tabla 10. Análisis de *casa.tif* para nivel 5 y estrategia *Balance dispersión -energía*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	86.37	98.57	98.58	24.82	214.47	21.02:1

Daubechies 5	117.94	98.96	98.94	24.47	232.46	21.18:1
Symlet 2	92.95	98.80	98.78	25.06	202.84	20.49:1
Coiflet 3	159.67	99.13	99.13	23.54	288.05	20.64:1

Imagen original

Imagen Reconstruida

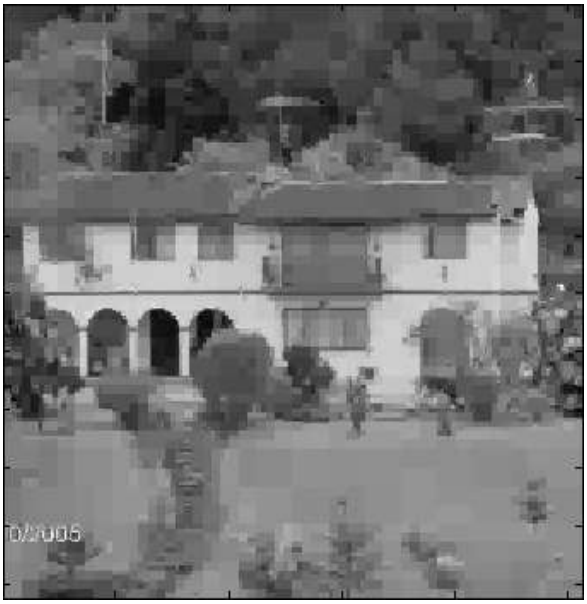

Figura 31. Imagen original y reconstruida de *casa.tif* para nivel 5 y estrategia *Balance dispersión – energía*

La imagen reconstruida de la figura anterior, puede ser catalogada relativamente como de baja calidad, juicio que por supuesto, depende de factores como el tipo de aplicación para la cual se requiera la imagen o sencillamente del concepto visual de cada persona. Adoptando un criterio de evaluación subjetivo fundamentado en la búsqueda de imágenes reconstruidas de 'calidad aceptable', se hace necesario establecer un compromiso entre las variables calidad de imagen y factor de compresión, lo cual tan solo se logra mediante la experimentación aplicando distintos umbrales a la imagen transformada. De este modo, la tabla 11 muestra los resultados obtenidos aplicando una estrategia que justamente por su naturaleza se ha denominado *criterio subjetivo*.

Tabla 11. Análisis de *casa.tif* para nivel 5 y estrategia *Criterio subjetivo*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	25	99.58	90.98	30.18	62.38	5.66:1
Daubechies 5	25	99.77	91.49	30.66	55.87	5.48:1
Symlet 2	25	99.66	91.89	30.50	57.98	5.93:1
Coiflet 3	18	99.91	86.58	32.60	35.76	3.67:1

La figura 32 presenta la imagen original y su imagen reconstruida correspondiente para la *wavelet Haar* de acuerdo al criterio en mención.

Figura 32. Imagen original y reconstruida de *casa.tif* para nivel 5 y estrategia *Criterio subjetivo*

De manera análoga, se realiza las mismas pruebas para la imagen *torre.tif* cuyos resultados se presentan a continuación.

Imagen: *torre.tif*
Peso en Bytes: 262506
Nivel de descomposición: 5

Tabla 12. Análisis de *torre.tif* para nivel 5 y estrategia *Remover cercanos a Cero*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	1.00	100	23.73	57.46	0.12	1.52:1
Daubechies 5	1.00	100	63.68	52.34	0.38	2.10:1
Symlet 2	1.00	100	64.93	52.28	0.38	2.28:1
Coiflet 3	1.00	100	63.28	52.30	0.38	1.92:1

Empleando ahora la estrategia *Balance dispersión – energía* se obtiene la información de la tabla 13, y la imagen reconstruida para la *wavelet Haar* de la figura 33.

Tabla 13. Análisis de *torre.tif* para nivel 5 y estrategia *Balance dispersión - energía*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	106.38	99.35	99.36	28.32	95.73	29.52:1
Daubechies 5	201.94	99.58	99.57	26.10	159.61	29.88:1
Symlet 2	127.37	99.47	99.48	27.79	108.25	28.34:1
Coiflet 3	326.92	99.57	99.57	24.67	221.87	25.75:1

Imagen original

Imagen Reconstruida

Figura 33. Imagen original y reconstruida de *torre.tif* para nivel 5 y estrategia *Balance dispersión – energía*

Finalmente, buscando establecer un compromiso entre calidad de imagen reconstruida y factor de compresión, se aplica un criterio subjetivo que entrega los resultados de la tabla 14.

Tabla 14. Análisis de *torre.tif* para nivel 5 y estrategia *Criterio subjetivo*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	7	99.97	89.13	42.65	3.54	5.35:1
Daubechies 5	8	99.99	89.57	42.02	4.08	5.07:1
Symlet 2	10	99.97	91.88	40.90	5.29	6.20:1
Coiflet 3	7	99.99	88.28	42.86	3.36	4.29:1

La figura 34 presenta la imagen original e imagen reconstruida correspondientes para la *wavelet Haar*.

Imagen original Imagen Reconstruida

Figura 34. Imagen original y reconstruida de *torre.tif* para nivel 5 y estrategia *Criterio subjetivo*

De acuerdo a la tabla 14 se tiene que el factor de compresión entre las dos imágenes anteriores es 6.20:1, en otras palabras, en términos del archivo que almacena la imagen comprimida significa que éste tiene un peso en Bytes resultante de aproximadamente 42340 Bytes frente a un peso de la imagen original de 262506 Bytes, lo cual puede considerarse como un buen factor de compresión considerando que se trata de imágenes de tamaño medio.

5.2 IMÁGENES DE DIMENSIONES GRANDES

No obstante los factores de compresión obtenidas por el sistema implementado para las imágenes *torre* y *casa* ambas de tamaño 512 X 512 píxeles y en formato TIF son un logro bastante significativo, más aún cuando el nivel de descomposición es alto, el verdadero rendimiento del sistema de compresión desarrollado se aprecia cuando la imagen que se desea comprimir es de tamaño y peso elevados. Por esta razón se presenta a continuación tres ejemplos con imágenes de tamaño mayor a 1024 píxeles por lado y con un peso alrededor de 3000000 Bytes (~ 3 MB), para finalmente comentar las medidas objetivas y la

percepción subjetiva producto de los resultados que se obtienen de dicho proceso de compresión.

Las imágenes a comprimir son: "*pájaro*", "*pueblito*" y "*facultad*" todas en formato TIF y con nivel de descomposición 5 ya que con este nivel fue con el que se obtuvo mejores resultados en cuanto a factores de compresión y calidad de imagen reconstruida. En cada caso, empleando un criterio subjetivo se muestra la imagen reconstruida para la cual se logra mejor factor de compresión.

Imagen: *pajaro.tif*
Peso en Bytes: 2915864
Nivel de descomposición: 5

Tabla 15. Análisis de *pajaro.tif* para nivel 5 y estrategia *Criterio subjetivo*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	30	99.93	96.58	34.77	21.67	11.54:1
Daubechies 5	100	99.77	99.06	28.50	91.80	25.46:1
Symlet 2	110	99.71	99.18	28.30	96.17	27.76:1
Coiflet 3	90	99.85	98.92	29.32	76.02	23.75:1

En la siguiente figura se muestra la imagen original y la reconstruida de "*pájaro*" para la *wavelet Symlet 2*, con la cual se logra un factor de compresión de 27.76:1 (peso de la imagen comprimida de aproximadamente 105038 Bytes).

Imagen original Imagen Reconstruida

Figura 35. Imagen original y reconstruida de *pajaro.tif* para nivel 5 y estrategia *Criterio subjetivo*

Imagen: *pueblito.tif*

Peso en Bytes: 2915136

Nivel de descomposición: 5

Tabla 16. Análisis de *pueblito.tif* para nivel 5 y estrategia *Criterio subjetivo*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	50	99.43	97.87	28.37	94.67	15.86:1
Daubechies 5	65	99.47	98.61	27.99	103.30	21.22:1
Symlet 2	70	99.35	98.84	27.56	114.13	23.80:1
Coiflet 3	65	99.57	98.54	28.11	100.52	20.59:1

La figura 36 muestra la imagen original y la reconstruida de "*pueblito*" para la *wavelet Symlet 2*, con la cual se logra un factor de compresión de 23.80:1 (peso de la imagen comprimida de aproximadamente 122485 Bytes).

Imagen original Imagen Reconstruida

Figura 36. Imagen original y reconstruida de *pueblito.tif* para nivel 5 y estrategia *Criterio subjetivo*

Imagen: *facultad.tif*

Peso en Bytes: 3697620

Nivel de descomposición: 5

Tabla17. Análisis de *facultad.tif* para nivel 5 y estrategia *Criterio subjetivo*

Wavelet	Umbral	% Energía	% Ceros	PSNR	MSE	FC
Haar	100	99.82	99.49	30.46	58.46	34.24:1
Daubechies 5	100	99.89	99.47	32.04	40.67	33.88:1
Symlet 2	100	99.87	99.52	31.78	43.20	34.36:1
Coiflet 3	110	99.90	99.48	31.77	43.27	32.37:1

La siguiente figura presenta muestra la imagen original y la reconstruida de "*facultad*" para la *wavelet Symlet 2*, con un factor de compresión de 34.36:1 (peso de la imagen comprimida de aproximadamente 107614 Bytes).

Imagen original Imagen Reconstruida

Figura 37. Imagen original y reconstruida de *facultad.tif* para nivel 5 y estrategia *Criterio subjetivo*

Es evidente que el desempeño de la implementación desarrollada para imágenes más grandes y de mayor peso es muy superior comparativamente al desempeño para imágenes medianas, basta con observar en las tablas anteriores los parámetros de evaluación objetivos, es decir, se obtienen unos factores de compresión elevados al mismo tiempo que se mantiene un buen valor para la PSNR.

Ahora bien, dado que no se puede obviar ni mucho menos desechar el criterio subjetivo, se observa que la calidad de las imágenes reconstruidas en estos últimos dos ejemplos es "muy buena". Más aún, si se desea establecer una relación entre el factor de compresión de estas imágenes y la calidad de las mismas, se podría afirmar con seguridad que el desempeño del algoritmo es también muy bueno.

De otro lado, si bien los resultados y el desempeño de la implementación tienen una buena calificación enmarcados claro está dentro del propósito general del proyecto, es importante mencionar que debido al análisis de la complejidad matemática que involucra cada proceso desarrollado se requiere de una gran potencia de cálculo, lo cual se revela en la alta complejidad computacional, máxime cuando la imagen a comprimir es de gran tamaño. Es decir, a mayor tamaño de imagen, mayor será el tiempo de procesamiento de la misma,

siendo éste en realidad el costo por obtener una buena compresión con una buena calidad final.

6 CONCLUSIONES Y RECOMENDACIONES

Este capítulo contiene las consideraciones finales del proyecto indicando en términos generales el desempeño del algoritmo desarrollado como herramienta software de ilustración de la compresión de imágenes con *wavelets*. Se plantea además algunas sugerencias en cuanto a trabajos futuros orientados al mejoramiento del algoritmo, así como en lo referente al campo de investigación de la *teoría wavelet* y sus aplicaciones.

6.1 CONCLUSIONES

Con el desarrollo del proyecto se ha diseñado e implementado un sistema de compresión de imágenes fijas con pérdidas basado en la *transformada wavelet*, el cual proporciona un entorno de experimentación en el campo de la compresión de imágenes y que esencialmente, ilustra una de las tantas aplicaciones de la *teoría wavelet.*

El algoritmo desarrollado implementa la *transformada wavelet discreta* (DWT) favoreciendo la compactación de energía y la generación de un gran porcentaje de *coeficientes wavelet* próximos a cero, los cuales son filtrados según un criterio de umbralación preestablecido o determinado por el usuario. El sistema se complementa con el algoritmo EZW como mecanismo de cuantificación para sacar provecho de la naturaleza jerárquica de la *descomposición wavelet*, y un módulo de codificación de entropía, para el cual se implementa un codificador aritmético que logra un buen rendimiento reflejado en buenos factores de compresión.

En cuanto al tipo de *familias wavelets* soportadas para realizar la transformación de las imágenes, es de resaltar que aunque no hizo parte de los objetivos del proyecto, el algoritmo posibilita la realización de un estudio comparativo de las más conocidas mediante la representación gráfica de la descomposición subbanda a distintos niveles, lo cual permite apreciar directamente el desempeño específico de cada familia en el filtrado digital de imágenes.

Por otro lado, con el propósito de evaluar los resultados proporcionados por el algoritmo y a la vez evaluar el desempeño del mismo, se cuenta con medidas objetivas que determinan la calidad de la imagen reconstruida respecto a la imagen original (criterios de calidad o fidelidad). Además, considerando que generalmente la percepción subjetiva de la calidad de una imagen entrega más información o es más significativa que cualquier medida de tipo numérico, el algoritmo presenta una gran flexibilidad a la hora de determinar qué tipo de información (coeficientes) de detalle se preserva de la imagen que se desea comprimir.

Es por supuesto el factor de compresión una medida clave para evaluar el desempeño del algoritmo implementado, por ende, mediante la experimentación con distintas variables de análisis tales como tipo de *wavelet*, nivel de descomposición, tipo y estrategia de umbralación, se han logrado resultados satisfactorios manteniendo medidas objetivas aceptables y degradación visual mínima, sobre todo en la compresión de imágenes de gran tamaño y peso.

Para calcular el factor de compresión, el algoritmo genera un archivo con extensión propia *.icw* que almacena los datos de la imagen comprimida resultantes del proceso de codificación aritmética de los coeficientes *wavelet* preservados. Desde luego, es este mismo archivo el argumento de entrada para el proceso de decodificación que termina con la representación de la imagen reconstruida.

En cuanto a las falencias del algoritmo se debe mencionar que el principal inconveniente radica en el tiempo de procesamiento producto de la intensidad computacional del módulo de cuantificación EZW, ello debido al concepto de aproximaciones sucesivas en el que se

fundamenta, claro está, éste varía dependiendo de las características del equipo (computador) donde el algoritmo se ejecute. Aunque esta situación no es tan crítica para comprimir imágenes de tamaño medio, sí se hace notable cuando se procesa imágenes de gran tamaño.

De cualquier forma, el algoritmo de compresión implementado cuenta con las características mínimas que debe poseer cualquier sistema de compresión que, como se demuestra en el trabajo desarrollado, no es cuestión solo de aplicar una transformada sino que requiere la implementación módulos adicionales como los de cuantificación y codificación. No se puede desconocer además que el desarrollo del proyecto ha implicado un estudio y análisis a fondo de la *teoría wavelet* y su aplicación en tratamiento digital de imágenes, fundamentalmente, en el campo de la compresión.

Adicionalmente, sin duda alguna uno de los mayores logros alcanzados con el desarrollo del proyecto consiste en la presentación de un estudio de la teoría *wavelet* que da a conocer de manera oficial a la Universidad del Cauca un concepto matemático relativamente novedoso, y que a su vez resalta la importancia de la aplicabilidad de dicha teoría, pues no en vano actualmente ésta es objeto de estudio en otras universidades a nivel nacional y en muchas otras a nivel mundial.

6.2 RECOMENDACIONES Y LÍNEAS FUTURAS

Aunque el algoritmo procesa imágenes de cualquier formato soportado por Matlab, presenta una limitación referente al tipo de imágenes, pues éste ha sido diseñado para procesar solo aquellas que son indexadas (imágenes compuestas por una matriz de datos y un mapa de color o *colormap*) y en escala de grises, de tal manera que no soporta imágenes a color (*truecolor*). De este modo, una extensión significativa al trabajo realizado en este proyecto puede plantearse como un sistema que permita comprimir toda clase de imágenes e incluso video.

Considerando la funcionalidad que el algoritmo ofrece, es posible realizar el *análisis wavelet* con distintas familias para estudiar los factores de compresión y criterios objetivos obtenidos con cada una. Además, dado que cualquier función que posea ciertas propiedades (duración finita y media cero) puede ser una función base para la *transformada wavelet*, resultaría interesante analizar qué características hacen que una función base sea propicia para la compresión de imágenes y determinar las propiedades de las imágenes para las cuales se adapta mejor.

En cuanto al proceso específico de cuantificación del algoritmo, se tiene cierta desventaja respecto a otro tipo de rutinas que hoy en día se hacen cada vez más populares en el ámbito del tratamiento digital de imágenes con *wavelets*. Dicha desventaja radica en que el EZW codifica los *coeficientes wavelet* según su magnitud independientemente de la subbanda a la que pertenezcan, de este modo, no es posible la decodificación de una determinada subbanda de manera independiente. Tal es el caso del SPIHT (*Set Partitioning In Hierachical Trees*), el cual se basa en el concepto del EZW en cuanto a la definición de las relaciones padre-hijos entre los *coeficientes wavelet* de diferentes subbandas, sin embargo, la codificación de los coeficientes se realiza de manera independiente para cada subbanda. Así mismo, otro método que opera de manera similar es el EBCOT (*Embedded Block Coding with Optimized Truncation*), método empleado en el estándar JPEG 2000.

No obstante, a pesar que la codificación EZW resulta más simple que los métodos mencionados anteriormente debido a que las operaciones que realiza se limitan a comparaciones y operaciones aritméticas, se constituiría en un gran avance el hecho de optimizar su funcionamiento.

De otra parte, uno de los campos más estudiados actualmente en cuanto a la aplicación de la *teoría wavelet* se refiere, es la implementación hardware de su transformada, con lo cual se busca solucionar en gran medida su inconveniente inherente en cuanto a tiempo de procesamiento y consumo de recursos computacionales; sin duda, es esta un área propicia para la investigación y experimentación que puede acarrear logros muy significativos.

Finalmente y considerando que en la Universidad del Cauca solo hasta ahora se empieza a difundir la *teoría wavelet*, resta incentivar el desarrollo de nuevos estudios referentes a la aplicación de la misma, pues son muchas las áreas donde puede representar importantes logros.

BIBLIOGRAFÍA

[1] Bultheel Adhemar. "Wavelets with Applications in Signal and Image Processing". Octubre de 2002.

[2] Kaiser Gerard. "A Friendly Guide to Wavelets". Birkhauser. Boston Basel Berlin. 1994.

[3] En línea: http://www.jpeg.org

[4] En línea:
http://www.mathworks.com/access/helpdesk/help/toolbox/wavelet/wavelet.shtml

[5] Faundez Pablo, Fuentes Alvaro. "Procesamiento Digital de Señales Acústicas utilizando Wavelets". Instituto de Matemáticas UACH. Chile.

[6] Hernández Díaz Marianito. Tesis profesional "Análisis Comparativo de Algoritmos para Reducción de Ruido en Señales Utilizando Wavelets". Universidad de las Américas, Puebla. Escuela de Ingeniería. Departamento de Ingeniería Electrónica. Cholula, Puebla, México a 11 de diciembre de 2003

[7] De Castro Fernández Rosa María. "Análisis de la Teoría de Ondículas Orientadas a las Aplicaciones en Ingeniería Eléctrica: Fundamentos". E.T.S.I. Industriales Departamento de Ingeniería Eléctrica. Madrid Julio de 2002.

[8] BURRUS, C. S., GOPHINATH, R., GUO, H. "Introduction to Wavelet and Wavelet Transforms, Prentice Hall, New Jersey, 1998.

[9] CHUI, C. K.,Wavelets: A Mathematical Tool for Signal Processing, SIAM, Filadelfia, 1997.

[10] En línea:
http://www7.nationalacademies.org/spanishbeyonddiscovery/mat_008276.html

[11] González Rafael C, Wood Richard E. "Tratamiento Digital de Imágenes". Addison-Wesley Iberoamericana, 1996.

[12] Maria Petrou. "Image Processing The Fundamentals". University of Surrey, Guildford, UK. Panagiota Bosdogianni. Technical University of Crete, Greece, 1999.

[13] En línea: http://www.bibliodgsca.unam.mx/tesis/tes7cllg/sec_18.htm

[14] Ash Robert B. "INFORMATION THEORY" Editorial New York : Dover Publications Inc 1965.

[15] En línea: http://www.arrakis.es/~patxy/compres.htm

[16] Cuello Rojas Fredy Fabián, Rueda Erazo Juan Carlos. "Compresión de Video Digital Bajo los Estándares MPEG". Fundación Universidad Autonoma de Colombia FUAC. En línea:
http://www.fuac.edu.co/autonoma/pregrado/ingenieria/ingelec/proyectosgrado/compresvideo/doc_b.htm

[17] En línea:
http://coco.ccu.uniovi.es/immed/compresion/descripcion/fundamentos/fundamentos.htm

[18] García Ramos Román. Tesis profesional "Compresión de imágenes fijas en MATLAB a través de DCT y WAVELET". Universidad de las Américas, Puebla. Escuela de Ingeniería. Departamento de Ingeniería Electrónica. Cholula, Puebla, México a 20 de enero de 2003.

[19] Fernández Marcos Martín. "Compresión de Imagen". 4 de mayo de 2004. En línea: http://lmi.bwh.harvard.edu/papers/ pdfs/2003/martin-fernandezCOURSE03f.pdf

[20] Gonzalo Pajares, Jesús M. de la Cruz. "Visión por Computador. Imágenes digitales y aplicaciones". Editorial RA-MA, España, 2001.

[21] Luong Chi Mai. "INTRODUCTION TO COMPUTER VISION AND IMAGE PROCESSING". Department of Pattern Recognition and Knowledge Engineering Institute of Information Technology, Hanoi, Vietnam, 2000.

[22] Molina Domínguez Eli Fernando. Tesis profesional "Sistema para compresión de imagenes en Matlab, a través de una serie de GUIs". Universidad de las Américas, Puebla. Escuela de Ingeniería. Departamento de Ingeniería Electrónica. Cholula, Puebla, México a 13 de mayo de 2004

[23] Fernández P. G, Ramírez J, Lloris A. "Procesamiento de Imágenes Utilizando la Transformada Discreta Coseno".
En línea: http://www.redeweb.com/microbit/articulos/9006863.pdf

[24] En línea: www.uhu.es/sergio.blanco/documentos/8.pdf

[25] Villaverde Vázquez Roberto. "Implementación y Prueba del Estándar JPEG 2000". Universidad de la Coruña. España. Enero de 2004.

[26] Fournier Natalia, Castro Gabriela, Russo Claudia. "Compresión de Imágenes Fijas utilizando la Trasformada Wavelet". Departamento de Informática. UNLP, Argentina.

[27] Martínez Ramírez Alejandro, Díaz Sánchez Alejandro, Linares Aranda Mónico, Vega Pineda Javier. "Onduletas y Fractales en la Compresión de Imagen y Video". INAOE Departamento de Electrónica, Tonantzintla Puebla, México. Instituto Tecnologico de Chihuahua, Chihuahua, México, 2002.

[28] Misiti Michel, Misiti Yves, Oppenheim Georges, Poggi Jean-Michel. “Wavelets toolbox for use with Matlab” User’s guide. Version 2. July 2002

[29] En línea:
http://coco.ccu.uniovi.es/immed/compresion/descripcion/spiht/ezw/ezw.htm

[30] Shapiro Jerome M. “Embedded Image Coding Using Zerotrees of Wavelet Coefficients” IEEE Transactions on signal processing. vol 41. No.12. December 1993

[31] Farrelle Michael Paul, "Recursive Block Coding for Image Data Compression". Springer-Verlag New York. 1990.

[32] Ian T. Young, Jan J. Gerbrands, Lucas J. van Vliet. "Fundamentals of Image Processing" Version 2.2. University of Technology. Netherland.

[33] “Matlab The language of technical computing”. Creating graphical user interfaces. Version 6.

[34] En línea:
http://perso.wanadoo.fr/polyvalens/clemens/download/ezwe.pdf

[35] En línea: http://www.fdi.ucm.es/profesor/alfredob/AEC2004/aritm_entera.pdf

[36] Morgan Kaufmann. "Computer architecture. A quantitative approach". Hennessy & Patterson. 1995.

[37] Flores Pompa Fátima, "Teorías y Aplicaciones de la Informática 2 - JPEG 2000" Universidad Católica "Nuestra Señora de la Asunción" Paraguay: Septiembre 2004.

[38] En línea: http://verona.fi-p.unam.mx/~boris/curso/apuntes/jpeg2000/jpeg2000.html