VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

FACULTAD DE INGENIERÍA

INGENIERÍA EN TELECOMUNICACIONES

Tesis de Licenciatura:

Transmisión Progresiva de Imágenes Digitales Codificadas para Redes de Datos, Utilizando la Transformada Wavelet y su Estructura de Árbol de Ceros

Alumnas:

Mauleón Fernández Julieta Isabel

Mondragón Gómez Miriam Verónica

Asesor:

Dr. Francisco Javier García Ugalde

México DF Junio 2007

A mis papas por darme el apoyo incondicional y el ejemplo, y a mis hermanas Adriana y Sofía por darme su consejo.

A Manuel Llamas por apoyarme, ayudarme y acompañarme estos años.

A Miriam Mondragón por ofrecerme su amistad y estar conmigo para lograr juntas esta importante meta en nuestras vidas.

A todos mis por acompañarme en el camino de la formación profesional y aportar algo en mi formación personal, en especial a Vernon Herrera, José Antonio Silva y a Ian Monsivais por ofrecerme también su ayuda en la realización de esta tesis.

Al Dr. Francisco Javier García Ugalde por guiarme en el tema del procesamiento digital de imágenes y darme su apoyo y consejo en la realización de esta tesis.

A los profesores de la facultad que como él, me motivaron para seguir estudiando e inculcaron en mí el amor a la ingeniería.

Julieta Isabel Mauleón Fernández

A mis padres por todo su apoyo y comprensión.

A mis hermanas Mónica y Liliana por todas la veces que me llevaron la cena en aquellas noches largas de estudio y por su apoyo a lo largo de mi vida.

A Ian por todo su amor, apoyo y tiempo.

A Julieta por haber sido una gran compañera y amiga a lo largo de la carrera así como también por haber hecho la tesis conmigo.

A los grandes amigos que hice en la Facultad, en especial a Vernon y José Antonio quienes siempre me han apoyado.

Al Dr. Francisco Javier García Ugalde por su apoyo en la elaboración de ésta tesis.

Miriam Verónica Mondragón Gómez.

Este trabajo fue realizado con apoyo parcial del proyecto:
SEP-CONACyT-41069.

Índice

Prefacio

En esta tesis se pretende explotar las propiedades de la transformada *wavelet* para imágenes digitales y de su estructura de árbol de ceros asociada, para diseñar un codificador progresivo que permita adaptarse dinámicamente al ancho de banda disponible de una red de datos, midiendo la calidad de las imágenes decodificadas de manera perceptiva.

Todo esto para lograr una compresión eficiente de imágenes y al mismo tiempo su transmisión progresiva, de tal forma que se pueda reconstruir la imagen, en el lado del receptor, ya sea con todos o sólo una parte de los datos que se envíen desde el transmisor pudiendo elegir la cantidad de datos a enviar.

En los siguientes capítulos se trataran los siguientes puntos:

Capítulo 1: En él se dará una introducción y se establece el planteamiento del problema que se pretende solucionar con la realización de esta tesis. Se da una breve introducción a los métodos de compresión existentes y se describen las ventajas de la utilización de la transformada wavelet como solución al problema planteado.

Capítulo 2: Comienza con la descripción de las imágenes digitales y da una visión más clara de los diferentes métodos de compresión, la teoría de la información y las formas de medir la calidad objetivamente. Con esta clasificación de los métodos de compresión de imágenes se escogen los que tengan las mejores propiedades desde el punto de vista de la tasa de compresión y de la razón señal a ruido.

Capítulo 3: Es una parte muy importante de la tesis ya que se da una explicación sobre lo que es en sí la transformada Wavelet Continua y sus propiedades. Se describe el análisis multiresolución de acuerdo a la función wavelet y la función escala, al igual que la Transformada Wavelet Discreta y el proceso para aplicarla a señales de dos dimensiones, como son las imágenes. El capítulo finaliza dando algunas aplicaciones de esta transformada.

Capítulo 4: Comprende la descripción detallada del proceso de codificación y cuantificación utilizado llamado EZW. Con este algoritmo se diseñó un codificador de imágenes digitales en base a lo estudiado y a las decisiones tomadas sobre su alcance respecto a esta tesis. También se da una descripción del formato comercial de compresión JPEG2000 para tener una referencia en los resultados.

Capítulo 5: En este capítulo se plasman los resultados de simulaciones y evaluaciones del codificador diseñado, mismos que se comparan con los resultados obtenidos de la norma JPEG2000. Gracias a estos resultados se obtienen una serie de conclusiones con respecto al cumplimiento del objetivo. A su vez, estas conclusiones dan lugar a diferentes trabajos que quedan pendientes para un futuro. También se expone una lista de referencias bibliográficas utilizadas para la elaboración de esta tesis.

CAPITULO I Introducción y planteamiento del problema

Introducción

Con el rápido crecimiento de Internet y la cada vez mayor demanda de transmisión de imágenes en ámbitos muy diversos (televisión digital, redes de computadores, telefonía móvil, televigilancia, etc.), se necesitan sistemas de compresión de imágenes que puedan brindar al usuario un servicio eficaz.

Además se necesitan sistemas de compresión de imágenes que puedan ser empleados para comprimir diferentes tipos de imágenes, como por ejemplo imágenes naturales, imágenes monocromáticas y en color, imágenes con diferentes niveles de gris, imágenes multicomponente, gráficos, imágenes realizadas mediante computador, texto, imágenes compuestas, imágenes médicas, etc.

La compresión de datos reduce el número de bits necesarios para una transmisión, o almacenamiento.

Debido a que la mayoría de las imágenes presentan semejanzas o correlaciones entre los valores de sus píxeles, podemos utilizar el valor de un píxel para predecir el de sus vecinos, de esta manera se puede comprimir imágenes reduciendo la cantidad de datos (píxeles) redundantes, conservando solamente los necesarios para representar de manera eficaz la información (imagen).

Existen varias técnicas de compresión de datos, según el tipo de éstos y la aplicación particular, se clasifican como técnicas de compresión con pérdidas y sin pérdidas.

La compresión sin pérdidas es utilizada en aplicaciones que no toleran ninguna pérdida de información, como son la edición de imágenes, la conservación de

material gráfico, las imágenes médicas, o los sistemas de medición mediante la imagen. La compresión con pérdidas, reduce la cantidad de información de una imagen, introduciendo un error, el cual no debe ser perceptible en la utilización de las imágenes.

Los sistemas de compresión de imágenes aprovechan las propiedades estadísticas de la información para conseguir la máxima compresión.

La mayoría de las técnicas de compresión de imágenes empleadas en la actualidad están basadas en transformadas. Estas transformaciones permiten distribuir de manera diferente la energía de una señal y son reversibles, es decir, conservan la energía. Al distribuir de manera diferente la energía, esta se concentra en unos cuantos coeficientes, permitiendo así la compresión.

Por sus propiedades de compactación de la energía, una de las transformaciones muy usadas es la “transformada coseno discreta” (DCT), que se utiliza en el estándar JPEG.

Existe una gran variedad de transformadas que pueden ser empleadas para la compresión de imágenes, una de las más recientes es “la transformada *Wavelet*”, la cual presenta unas características de análisis muy apropiadas para ser empleada en compresión de imágenes.

En la actualidad existe una gran variedad de sistemas de compresión de imágenes, y para diferentes aplicaciones se han desarrollado diferentes estándares por parte de los organismos internacionales de estandarización.

Planteamiento del problema a resolver en esta tesis

Contexto y justificación

La compresión de imágenes digitales es un problema siempre presente en la transmisión a través de redes de computadoras y en el almacenamiento en memoria, por el costo que tiene el uso del ancho de banda y el alto volumen de información que representan las imágenes digitales.

La búsqueda y validación de nuevos métodos de compresión, que permitan alcanzar altas tasas de compresión y una alta razón señal a ruido de las imágenes procesadas, es y seguirá siendo un reto. Esto último debido al alto costo de las comunicaciones y de las memorias, así como al aumento constante de cantidad de información a enviar o almacenar.

Alcance

Con esta tesis se pretende por un lado probar una de las propiedades que tienen las transformadas *wavelets* y que es que, en una estructura de descomposición multirresolución, los valores de los coeficientes de las transformadas presentan cierta correlación que se puede asimilar a un árbol. En ese árbol, cuando un nodo de un nivel n, llamado “padre”, es de valor pequeño, es muy probable que sus nodos del nivel n-1, llamados “hijos”, sean también de valor pequeño. Por lo que se puede explotar esta propiedad para propósitos de compresión.

Por otro lado, una vez que se establece este tipo de estructura de arbol, la transmisión puede ser progresiva (envío inicial de la información más importante) y de varios niveles de tasa de transmisión. Esto es porque una vez estructurados los coeficientes en forma de árbol, se pueden enviar primeramente los más significativos. Entonces se tiene la libertad de escoger la tasa de transmisión que se desea, siempre procurando mantener la mejor razón señal a ruido. Esta

propiedad es muy útil en redes de computadoras donde el nivel de ruido es variable y por ende también el ancho de banda disponible.

Se pretende también medir la calidad de las imágenes descomprimidas de manera perceptiva y utilizar esta medida para ajustar la tasa de transmisión.

Relevancia

- Evaluación de la eficiencia y robustez de un método de compresión de imágenes digitales dentro del marco de una codificación multirresolución, de manera similar a la especificada en la norma de codificación JPEG-2000 basada en *wavelets*.

CAPÍTULO II Fundamentos del procesamiento de imágenes

Para comprimir imágenes es fundamental conocer la manera en que las imágenes digitales se encuentran conformadas, por ello en este capítulo se da una breve explicación de éstas y de sus propiedades principales.

También se brinda una amplia explicación de los diferentes métodos de compresión, de la teoría de la información y de las medidas de calidad.

Además se habla de las transformadas más comunes que se ocupan para el procesamiento de las imágenes y se hace una explicación amplia de la transformada *wavelet* discreta, ya que ésta es la transformada que se utilizo para el desarrollo de esta tesis.

2.1 Imagen digital

Se puede definir a una imagen como una función bidimensional *f(x,y)*, donde *x* e *y* son las coordenadas espaciales, y la amplitud de *f* en cualquier par de las coordenadas se denomina como la intensidad, o nivel de gris, o de color, de la imagen en el punto con coordenadas (x,y).

Si los valores de la amplitud de *f(x,y)* son todos finitos, entonces se trata de una imagen digital, que cuenta con valores discretos.

Una imagen digital de dimensiones MxN se encuentra conformada de un número finito de elementos, cada uno de ellos tienen un valor particular y localización. En la literatura técnica estos elementos se llaman elementos unidad de imagen, o píxeles. La imagen digital es entonces un arreglo rectangular, ver Figura 2.1.1.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & ..... & f(0,N-1) \\ f(1,0) & f(1,1) & ..... & f(1,N-1) \\ . & . & ..... & . \\ f(M-1,0) & f(M-1,1) & ..... & f(M-1,N-1) \end{bmatrix}$$

Figura 2.1.1.Arreglo rectangular.

Los píxeles son una medida de división de celdas, por lo que no se puede decir que un píxel mide 1[cm^2] o 1[km^2], entonces cuando se habla de una imagen de M x N píxeles no se sabe su tamaño real ni físico, sólo se sabe que la imagen esta dividida en M x N celdas figura 2.1.2.

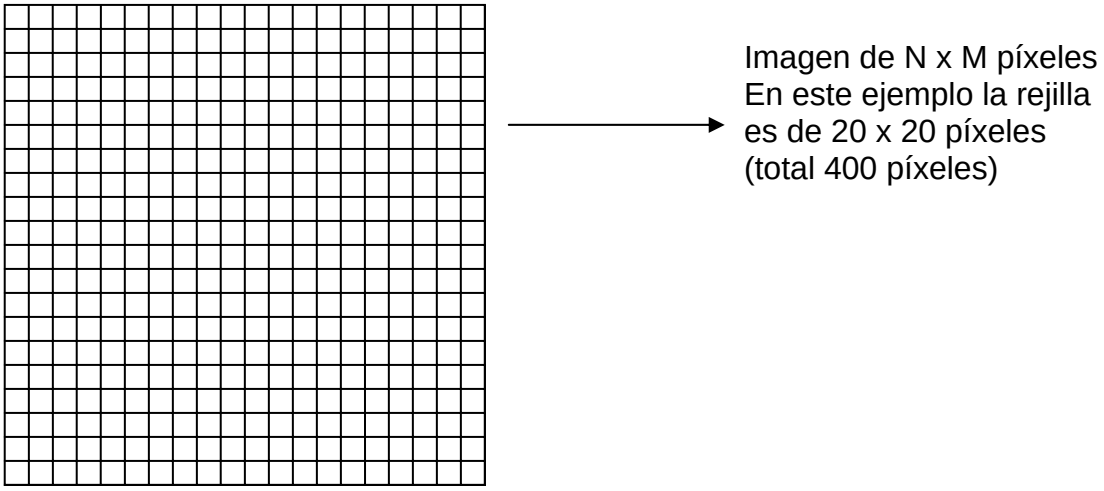


Figura 2.1.2.División de celdas.

Sin embargo, al asignarle a la imagen una resolución (es decir, qué tanto detalle tiene la imagen), entonces sí es posible saber de qué tamaño es la imagen, lo que implica entonces que una imagen de alta resolución dará como resultado un archivo de mayor tamaño que la misma imagen a una resolución inferior.

Debido a la naturaleza binaria del procesamiento numérico, cuando contamos con imágenes que son matrices cuadradas NxN (donde N es una potencia de dos), como por ejemplo imágenes de 512x512, o de 256x256 etc., los procedimientos matemáticos que se llevan a cabo para procesarlas se simplifican, es por eso que en este trabajo se utilizaran matrices cuadradas que sean potencias de 2.

El número de bits “b” que se requieren para guardar una imagen de tamaño NxN con 2^m diferentes niveles de grises es:

$$b=N \times N \times m \qquad (2.1.1)$$

Las imágenes digitales son muy prácticas, ya que las podemos visualizar en la computadora, las podemos imprimir, guardar en un dispositivo de almacenamiento, como un CD, además de que se pueden transmitir.

Sin embargo para que las imágenes sean almacenadas, se requiere la utilización de los archivos de computadora, que son una manera de codificar la información.

Para las imágenes, se han definido varios archivos que siguen determinados formatos y sus características principales se encuentran resumidas en la tabla 2.1.1.

Formato	**Compresión / Tipo**	**Profundidad de color**	**Uso típico**
TIFF	Opcional / Sin pérdidas	1 a 64 bits	Imágenes de alta calidad, cámaras digitales, escáneres, impresión
RAW	Sin pérdidas	48 bits	Cámaras digitales
JPEG	Con pérdidas	8 o 24 bits	Cámaras digitales, Internet, impresión, intercambio de imágenes
GIF	Sin pérdidas	1 a 8 bits	Internet, imágenes de reducido tamaño, logos...
PNG	Sin pérdidas	1 a 48 bits	Internet, gráficos, iconografía software
PSD	Sin pérdidas	1 a 64 bits	Edición y manipulación
PCX	Sin pérdidas	1 a 24 bits	Formato de PC Paintbrush
BMP	Sin pérdidas	1 a 32 bits	Desarrollado por Microsoft como formato de mapa de bits nativo del sistema operativo Windows.
PGM	Con o sin pérdidas.	1 a 8 bits	Internet
JPEG2000	Con o sin pérdidas	1 a 24 bits	Internet, aunque no es soportado por todos los navegadores.

Tabla 2.1.1. Características principales de los formatos.

Existen varios tipos de imágenes, pero a grandes rasgos se pueden dividir en dos grandes grupos:

- **Imágenes raster o vectoriales:**

  En este tipo de imágenes, la información de cada uno de los puntos se recoge por medio de un modelo matemático que lo relaciona con el resto de los puntos que forman la imagen.

  Dado a que un modelo matemático, no ocupa mucho espacio y es suficiente para representar una curva en su totalidad, es el tipo adecuado para el diseño de línea y figura.

- **Imágenes de mapa de bits o bitmap:**

  Este tipo de imágenes se construyen describiendo cada uno de los puntos que componen la imagen, por lo que entonces se tiene información específica de cada uno de ellos, como su posición y su color. Dentro de este tipo se encuentran muchos formatos, algunos que son soportados directamente por los navegadores.

2.1.1 Clasificación de las imágenes con respecto al número de bits por píxel:

Las imágenes tienen cierta profundidad de intensidad, que es una propiedad que se refiere al número de tonos diferentes que se pueden visualizar en una misma imagen. Por lo tanto cada píxel se puede representar con un diferente número de bits.

Debido a la naturaleza del sistema binario de numeración, una profundidad de bits de n implica que cada píxel de la imagen puede tener 2n posibles valores y por consiguiente, se representan 2n tonos distintos.

Debido a la aceptación prácticamente universal de los grupos de 8 bits como unidades básicas de información en los dispositivos de almacenamiento, los valores de profundidad de intensidad suelen ser divisores o múltiplos de 8, como

lo son 1, 2, 4, 8, 16, 24 y 32, con la excepción de la profundidad de intensidad de 15, que es usada por algunos dispositivos gráficos.

2.2 Fundamentos de la compresión de imágenes y teoría de la información

La compresión de imágenes es una área de investigación para crear archivos de tamaño manejable y de dimensiones aptas para transmitir rápidamente, además de que permite una disminución en el ancho de banda necesario para la transmisión o el almacenamiento.

Se puede realizar la compresión de una imagen sin una degradación significativa de la calidad visual, siempre y cuando la imagen tenga en alto grado en una de las siguientes cantidades:

a) Redundancia espacial, debido a la correlación entre los píxeles circundantes,
b) Redundancia espectral, debido a la correlación entre los componentes frecuenciales, y
c) Redundancia Psicovisual, debido a las propiedades del sistema visual humano, el cual percibe de manera diferente las regiones uniformes de las regiones texturizadas.

En general, entre mayor redundancia exista, se puede alcanzar una mayor compresión.

Los tipos de redundancia antes mencionados se pueden agrupar en dos grupos: redundancia estadística (espacial) y psicovisual. La redundancia espacial está presente cuando ciertos patrones espaciales se parecen a otros. Mientras que la psicovisual, es debida al sistema de percepción, el cual es insensible a ciertas frecuencias espaciales.

En la figura 2.2.1 se muestra un diagrama general de bloques de un sistema de compresión de datos:

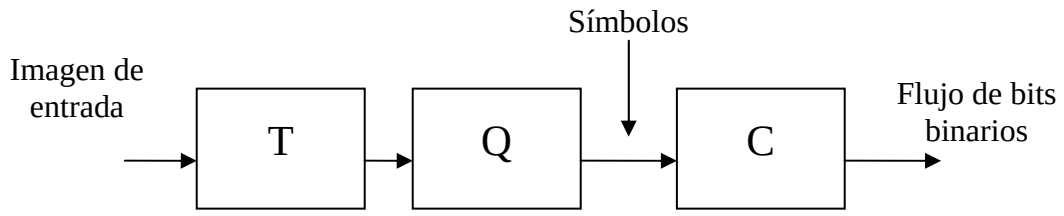


Figura 2.2.1 Esquema de principio de un sistema de compresión

Los bloques constitutivos del sistema general mostrado en la figura 2.2.1, corresponden a:

Transformación:

El bloque marcado con la letra (T) aplica una **transformación** uno a uno a los datos de la imagen de entrada. La salida de este bloque es un conjunto de coeficientes, donde la mayor parte de la energía queda concentrada en unos cuantos, de tal forma que su compresión sea más eficiente que con la imagen original.

Las Transformadas matemáticas son aplicadas a las señales para obtener una distribución diferente de la energía, y para poder interpretar las señales vistas desde un espacio diferente. Así, después transformar una señal por medio de cualquier transformada matemática disponible, se dice que es una señal **procesada**. Las transformadas deben ser reversibles para poder regresar al espacio original.

Prácticamente, la mayoría de las señales que percibimos, se encuentran en el **dominio del tiempo**, esto quiere decir que a cada instante de tiempo le corresponde cierta amplitud o valor. Cuando representamos estas señales, tenemos dos ejes: el del tiempo y el de la amplitud. Esta representación no siempre es la mejor para aplicaciones relacionadas con el procesamiento. En

muchos casos, la información que nos interesa está en la frecuencia contenida en la señal, y para su estudio se requiere de la Transformada de Fourier.

Como se mencionó anteriormente, existen muchas transformadas que pueden ser aplicadas a las señales para obtener de ellas cierta información. Posiblemente, la más conocida sea la Transformada de Fourier (FT). Como es sabido, si se aplica la FT a una señal en el dominio del tiempo, se obtiene su representación en el dominio de la frecuencia. Esta representación nos indica como se distribuye la energía de la señal en el dominio de la frecuencia. Se obtiene entonces el espectro de frecuencia de una señal, que básicamente representa los componentes en frecuencia de la señal (componentes espectrales). La figura 2.2.2 ilustra estas componentes para una señal sintética que contiene dos frecuencias.

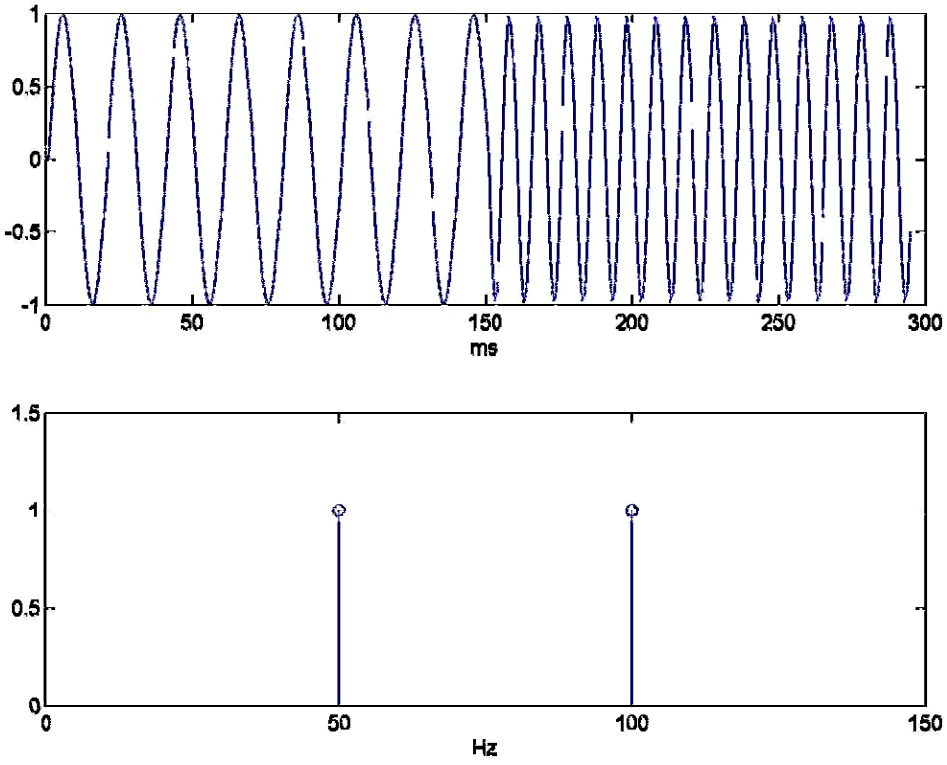


Figura 2.2.2. Señal que cambia de frecuencia a los 150 ms en el dominio del tiempo (arriba) y de la frecuencia (abajo)

Por sus propiedades, en la compresión de imágenes se usan otras transformadas como son la Transformada Coseno Discreta, la cual empaqueta la energía de la señal en un pequeño número de coeficientes; más recientemente se tiene el mapeo de multiresolución, tal es el caso de la descomposición en subbandas y la Transformada *Wavelet*.

Cuantizador:

El bloque (Q) representa el proceso de **cuantización**, el cual genera un número limitado de niveles de gris que pueden ser usados en la representación de la imagen comprimida. La cuantización es un mapeo de varios-a-uno, el cual puede ser reversible, pero por lo general no lo es. Un buen cuantizador es aquel que representa la señal original con mínimas pérdidas y distorsión (mínimo error de cuantificación).

Existen dos tipos de cuantizadores, los escalares y los vectoriales. Los primeros se refieren a una cuantización de datos elemento por elemento, mientras que los segundos cuantizan un bloque de datos, llamado vector, a la vez.

Codificador:

El bloque (C) representa el **codificador**, el cual asigna una palabra codificada, es decir, una cadena binaria, a cada símbolo a la salida del cuantizador. El codificador puede emplear códigos de longitud fija, o variable. Los de longitud variable (VLC por sus siglas en inglés), también conocidos como **codificadores de entropía**, utilizan palabras codificadas de tal forma que se minimiza la longitud promedio de la representación binaria de los símbolos. Esto se logra asignando una palabra codificada más corta a los símbolos con mayor probabilidad de ocurrencia, que es el principio fundamental de la codificación de entropía.

Diferentes sistemas de compresión de imágenes implementan diferentes combinaciones de estas opciones, generando entonces codificadores híbridos. De manera general los métodos de compresión pueden ser clasificados en:

a) **Lossless** o sin pérdidas, cuyo objetivo es minimizar la tasa de bits sin distorsión en la imagen.
b) **Lossy** o con pérdidas, cuyo objetivo es obtener la mejor fidelidad posible para una tasa de bits dada, o minimizar la tasa de bits para alcanzar una medida de fidelidad dada.

Los bloques de transformación y de codificación son sin pérdidas, mientras que el de cuantización es con pérdidas, por tanto, un método de compresión sin pérdidas que hace uso de la redundancia estadística, no utiliza un cuantizador, o bien codifica también el error de cuantificación para poderlo sumar en la decodificación. En muchos casos prácticos, se permite una pequeña degradación (de preferencia imperceptible) en la imagen para alcanzar la tasa de bits deseada.

Los métodos de compresión con pérdidas, hacen uso de la redundancia estadística y psicovisual. Estos métodos de compresión se analizarán con mayor detalle más adelante.

Ahora que se ha descrito de manera general la compresión es necesario considerar algunos aspectos de la **teoría de la información**, la cual es una rama de la teoría matemática de la probabilidad y la estadística que estudia la información y todo lo relacionado con ella: canales, compresión de datos, criptografía, etc.

Fue establecida por Claude E. Shannon a través de un artículo publicado en el *Bell System Technical Journal* en 1948, titulado “A mathematical Theory of communication” (*Una teoría matemática de la comunicación*).

La Teoría de la Información se desarrolla en términos de probabilidades, ya que la información tiene una naturaleza aleatoria.

2.2.1 Modelo de comunicación

El modelo de comunicación desarrollado por Shannon y Weaver se basa en un sistema de comunicación general que puede ser representado como se muestra en la figura 2.2.3.

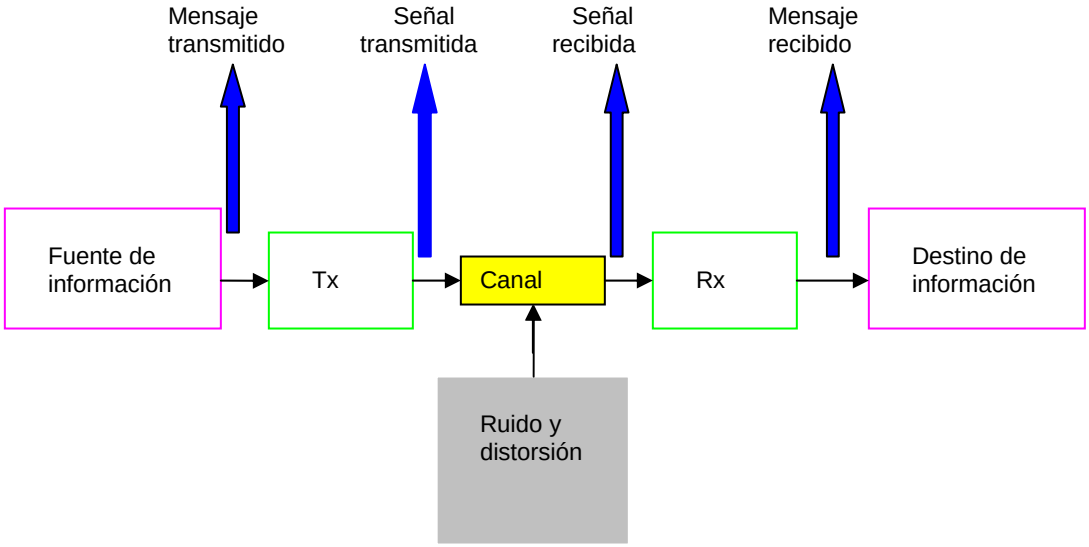


Figura 2.2.3. Esquema general de un sistema de comunicaciones

Se puede ver que el primer elemento del sistema es la **fuente de información**, la cual en este trabajo se trata de imágenes digitales.

El siguiente elemento es la **transmisión de información**, ya sea voz, video, cartas, periódicos, libros, etc., el transmisor transforma el mensaje de alguna manera en algún tipo de señal que pueda ser transportada por el canal.

El **canal** es el medio usado para transmitir alguna señal del transmisor al receptor. Al viajar la señal por el canal, la señal puede ser alterada por ruido o distorsionada.

Como siguiente elemento se tiene al **receptor**, que trabaja con la señal transmitida para poder reproducir el mensaje original.

Finalmente se tiene el **destino de la información**, que es en donde se obtiene el mensaje original.

La información es algo que sí se puede cuantizar, una cantidad de información tiene un valor numérico útil.

Es importante poder diferenciar entre datos e información, aunque muchas veces se tomen como sinónimos, no lo son, los datos pueden o no contener información. Los datos son una forma de representar la información; así, una misma información puede ser representada por distintas cantidades de datos. Por tanto, algunas representaciones de la misma información contienen datos redundantes.

Una forma común de medir el contenido de la información en un mensaje es con la **entropía**. La entropía es una magnitud que nos da el grado de desorden, o de incertidumbre de un sistema. La información tiene carácter aleatorio, ya que la contenida en un símbolo está relacionada con qué tan imprevisto, o inesperado, sea ese símbolo.

Si es seguro que ocurra un evento, la probabilidad matemática de que éste ocurra, por definición, es uno. Si es transmitido un mensaje de que un evento de este tipo ocurrió, no hay información alguna, ya que nosotros lo sabíamos con probabilidad de ocurrencia uno incluso antes de recibir el mensaje.

Si un evento es poco seguro que ocurra, entonces su probabilidad de ocurrencia es muy baja. Cuando es transmitido un mensaje acerca de que un evento de tal naturaleza ocurrió, entonces el mensaje tiene mucha información, ya que, de no ser por él, no sabríamos de la ocurrencia del evento.

Por último, si no existe seguridad de que un evento ocurra o no, entonces la probabilidad de ocurrencia tiene un valor entre cero y uno. Un mensaje transmitido acerca de la ocurrencia de este tipo de evento, va a contener información, sin embargo la cantidad de esta depende de la probabilidad de ocurrencia del mensaje.

Por lo tanto, de acuerdo a la teoría de la información de Shannon, definiremos la cantidad de información recibida en el mensaje como [2]:

$$\text{Información recibida} = \log\left(\frac{\text{Probabilidad en el receptor de que un evento ocurra después de que el mensaje es recibido}}{\text{Probabilidad en el receptor de que un evento ocurra antes de que el mensaje es recibido}}\right) \quad (2.2.1)$$

La probabilidad del numerador, depende de la confiabilidad del mensaje. Si estamos seguros de que el mensaje recibido es correcto, es decir, el caso en el que el mensaje no existe ruido, entonces la ecuación se podría simplificar a:

$$\text{Información recibida} = -\log\left(\text{Probabilidad en el receptor de que un evento ocurra antes de que el mensaje es recibido}\right) \quad (2.2.2)$$

En la práctica, la probabilidad de ocurrencia de cada símbolo es estimada con el histograma normalizado de la fuente específica.

La entropía H(X) de una fuente discreta de información DMS (*Discrete Memoryless Source*) *X* con un alfabeto *A* está definida como el promedio de información por símbolo en la fuente, dado por:

$$H(X) = \sum_{a \in A} p(a) \log_2\left(\frac{1}{p(a)}\right)$$

$$H(X) = -\sum_{a \in A} p(a) \log_2(p(a)) \quad (2.2.3)$$

Entre menos incertidumbre haya en la ocurrencia de los símbolos, menor será la entropía de la fuente. La entropía es máxima para una distribución uniforme, que se da cuando todos los símbolos tienen la misma probabilidad de ocurrencia (mayor incertidumbre).

Por ejemplo, suponiendo una imagen de 8 bits por píxel como la realización de una DMS *X*, cuyos símbolos *i* son los niveles de gris de los píxeles y el alfabeto *A* es la colección de todos los niveles de gris entre el 0 y el 255, entonces la entropía de la imagen será:

$$H(X) = -\sum_{i=0}^{255} p(i)\log_2(p(i)) \tag{2.2.4}$$

Donde $p(i)$ denota la frecuencia relativa de ocurrencia del nivel de gris *i* en la imagen. Nótese que si la imagen esta constituida por un solo nivel de gris, la entropía sería cero, dado que no hay incertidumbre.

Una metodología para modelar las fuentes que toma en cuenta la correlación entre los valores vecinos de los símbolos es con el proceso aleatorio de Marcov-K de orden K. Este describe la probabilidad de ocurrencia de un símbolo, dependiendo del valor de K símbolos precedentes. En esta tesis consideraremos el modelo de la fuente como una fuente discreta sin memoria DMS.

2.3 Métodos de compresión de imágenes

Como se mencionó anteriormente, en general existen dos categorías para los algoritmos de compresión de datos: sin pérdidas (*lossless*) y con pérdidas (*lossy*).

Las técnicas de compresión con pérdidas causan una degradación en la calidad de la imagen en cada paso de compresión y descompresión. Sin embargo, considerando la percepción visual humana, podemos asegurar una degradación imperceptible ya que ésta depende de la tasa de compresión que seleccionemos. Generalmente la compresión con pérdidas ofrece una mayor tasa de compresión que la compresión sin pérdidas.

2.3.1 Compresión sin pérdidas (lossless)

Una compresión sin pérdidas garantiza que la imagen descomprimida será exactamente igual a la imagen antes de comprimirla. Este es un requisito importante para algunas áreas de aplicación en donde no sólo la alta calidad es importante sino que también la no alteración del archivo es un requisito legal.

Algunos de los métodos de compresión sin pérdidas son por ejemplo:

- **RLE (Run Length Encoding)**
- **Codificación de Huffman**
- **Codificador de entropía (Lempel/Ziv)**
- **Codificación de área**

2.3.2 Compresión con pérdidas (lossy)

En muchas aplicaciones (por ejemplo televisión digital) no es necesaria una reconstrucción exactamente igual a la imagen original, lo que nos permite una compresión más eficiente, y esto lo logramos con la compresión con pérdidas. Estas técnicas de compresión generalmente tienen tres componentes:

- *Modelado de imagen:* define lo que se le va aplicar a la imagen, como es por ejemplo una transformación.
- *Cuantización de parámetros:* donde los datos generados de la transformación son cuantizados para reducir la cantidad de información.
- *Codificación:* donde un código es generado asociando las palabras código apropiadas a los datos producidos por el cuantizador.

Cada uno de estos pasos contribuye a la compresión. El modelado se aplica para poder explotar características estadísticas de la imagen, como es la correlación, o la redundancia. Ejemplos comunes son los métodos de transformación, que como ya se han mencionado anteriormente, representan la imagen en un dominio distinto. Ejemplos de transformadas son la Transformada Coseno Discreto, la

Transformada Kahrunen-Loeve, la Transformada *Wavelet*, entre otras, donde un número reducido de coeficientes contienen la mayor parte de la información de la imagen original. En muchos casos, esta primera fase no implica una pérdida de información, es decir, la transformada es reversible.

Como se ha visto, la cuantización consiste en reducir la cantidad de datos para representar la información en el nuevo dominio. Como en la mayoría de los casos la cuantización no es reversible, por tanto pertenece a los métodos de compresión con pérdidas.

La codificación estadística está generalmente libre de errores y optimiza la representación de la información.

En la siguiente sección se detallarán algunas de las transformadas mencionadas que se utilizan en la compresión con pérdidas.

2.4 Codificación por transformación

La codificación por transformación es una herramienta que tiene diversas aplicaciones, y por lo tanto es usada en distintas áreas.

Algunas de sus aplicaciones en el campo de la informática son:

- Procesamiento de voz.
- Reconocimiento de caracteres.
- Identificación y verificación de firmas.
- Caracterización de secuencias binarias.
- Análisis y diseño de sistemas de comunicaciones.
- Filtrado digital.
- Análisis espectral.
- Compresión de imágenes.
- Compresión de datos.

- Procesamiento de señales.
- Análisis multifractal.

Las transformadas discretas sientan las bases en los estándares mundiales para la compresión con pérdidas.

Un codificador de transformada básico segmenta a la imagen en diferentes bloques, cada bloque experimenta una transformación bidimensional ortogonal, de donde se obtiene un arreglo de coeficientes transformados, después cada coeficiente transformado es cuantizado y codificado.

Los coeficientes que poseen mayor energía son cuantizados más finamente, y aquellos que poseen menor energía son cuantizados no tan finamente, o son simplemente truncados.

El codificador trata a los coeficientes cuantizados como símbolos y los codifica por medio de un codificador de entropía. El decodificador reconstruye las intensidades de los píxeles a través de la cadena recibida, aplicando el proceso inverso al codificador.

La mejor transformada estadísticamente hablando, es aquella que descorrelaciona a los coeficientes de la imagen transformada, empaqueta la mayor energía (información) en pocos coeficientes, minimiza el error cuadrático medio (MSE) entre la imagen reconstruida y la original, y también minimiza la entropía total comparada con otras transformadas.

La transformada que satisface las condiciones anteriores es la transformada Karhunen-Loêve (KLT), pero no es utilizada debido a que:

- Sus funciones de base dependen de la matriz de covarianza de la imagen, lo que implica que para cada imagen se debe de procesar de nuevo y transmitir.

- La descorrelación perfecta no es posible, debido a que las imágenes naturales no se pueden modelar como realizaciones de campos aleatorios homogéneos.
- No existen algoritmos rápidos que permitan su implementación.

Existen otras transformadas que sí se utilizan en la compresión de imágenes, como son la Transformada Coseno Discreta (DCT), y la Transformada Discreta *Wavelets* (DWT) entre otras.

A continuación se explicarán brevemente estas transformadas:

2.4.1 Transformada Coseno Discreta

Es una transformada ortogonal que posee un algoritmo eficiente, es la clave en el estándar de compresión JPEG, está basada en la Transformada Discreta de Fourier, pero sólo descompone a la imagen en sumas de cosenos, porque sus funciones de base son cosenoidales.

Formalmente, la Transformada de Coseno Discreta es una función lineal e invertible F: $R_n \rightarrow R_n$ (donde R es el conjunto de números reales), o, de forma equivalente, una matriz cuadrada n × n.

Dependiendo de cómo se definan unas constantes de normalización para que las funciones de base tengan energía unitaria, existen diferentes variantes de la DCT. Pero en general n números reales $x_0, ..., x_{n-1}$ se transforman en los n números reales $f_0, ..., f_{n-1}$ de acuerdo a la siguiente ecuación :

$$f_j = \sum_{k=0}^{n-1} x_k \cos\left[\frac{\Pi}{n} j\left(k + \frac{1}{2}\right)\right] \qquad (2.4.1)$$

2.4.2 Transformada Discreta *Wavelet*

La trasformada de ondeletas (*wavelets*) es la representación de una señal como una combinación lineal de unas funciones de base más apropiadas. Particularmente útil para señales no estacionarias, produciendo un mejor compromiso entre la resolución temporal y frecuencial.

La transformada discreta *wavelet* se utiliza para la codificación computacional de señales. Entre sus distintas aplicaciones se pueden mencionar:

- Compresión de señales e imágenes.
- Análisis del electroencefalograma.
- Compresión de imágenes de huellas dactilares.
- Detección de ondas sísmicas.
- Detección de cambios en estructuras y señales, entre otras.

Para esta transformada se usa una función llamada *wavelet* , la cual es integrable y oscilatoria con media nula.

$$\int \Psi(t)dt = 0 \qquad (2.4.2)$$

Las *wavelets* son funciones matemáticas que separan la información en distintas componentes frecuenciales, y cada componente es estudiada con una cierta resolución asociada a la escala.

La familia de wavelets se define como:

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}}\Psi\left(\frac{t-b}{a}\right);\ a,b\ \varepsilon\ R, a>0 \qquad (2.4.3)$$

Donde a es el parámetro de escalamiento, o dilatamiento, y b es el parámetro de desplazamiento, o traslación.

Esta familia de funciones son una copia de una función prototipo Ψ(t), que se conoce como **wavelet madre,** trasladada y escalada mediante las variables b y a.

En la teoría de *wavelets* existen muchos conceptos importantes, como lo son los momentos de desvanecimiento, soporte compacto y la simetría.

Los **momentos de desvanecimiento** nos permiten conocer la forma de la *wavelet*, y se encuentran definidos por:

$$\int_{-\infty}^{\infty} \Psi(t) t^i dt = 0 \qquad (2.4.4)$$

De lo anterior se puede determinar que una función tiene v momentos de desvanecimiento si la integral es 0 para i=0,1,….,v-1.

Generalmente el número de momentos de desvanecimiento de una *wavelet* determina el orden de la transformada *wavelet*.

El **soporte compacto** de una *wavelet* se refiere a que las funciones base no son cero en un intervalo finito, lo que se define como:

$$\Psi(t) = 0, \text{ si } |t| > N, \text{ para alguna } N \qquad (2.4.5)$$

La **simetría** en los filtros se busca con el fin de evitar distorsiones en la información mediante la fase lineal, esto se expresa en la siguiente ecuación donde k es una constante y **w** es la fase.

$$\Psi(w) = kw. \qquad (2.4.6)$$

Existen muchas familias de *wavelets*, las cuales se dividen en dos grupos principales, estos son:

- *Wavelets* Ortogonales.
- *Wavelets* Biortogonales.

Entre las *wavelets* ortogonales se tiene a la familia definida por Daubechies, la cual se representa con la notación 'dbN', donde N indica el orden y $N\ \varepsilon\ Z$.

Esta *Wavelet* posee soporte compacto y N momentos de desvanecimiento, puede ser ortogonal, biortogonal y no posee simetría.

La wavelet db1 (Haar) es una de las más sencillas porque sus funciones de base son constantes por tramos y a pesar de que posee soporte compacto, no cuenta con una buena localización tiempo-frecuencia.

La apariencia de las *wavelets* de Daubechies se muestra en la figura 2.4.1:

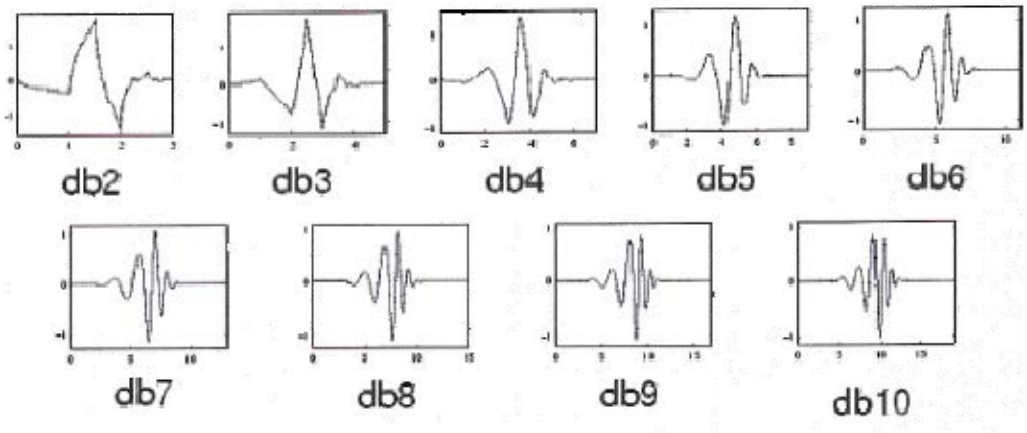


Figura 2.4.1. Apariencia de las *wavelets* de Daubechies

Para agregar un cierto grado de simetría a estas *wavelets*, Daubechies creo a la familia Symmlets.

Las wavelets biortogonales poseen soporte compacto y simetría, así como la reconstrucción perfecta de la señal con posibles filtros FIR (Finite Impulse Reponse), lo cual en las *wavelets* ortogonales es imposible, con excepción de la *wavelet* Haar.

El orden de estas *wavelets* esta dado por los parámetros Nr y Nd, para la reconstrucción y para la descomposición, respectivamente.

Los momentos de desvanecimiento de $\Psi(t)$ están dados por Nr.

La apariencia de algunas waveles biortogonales se aprecia en la figura 2.4.2.

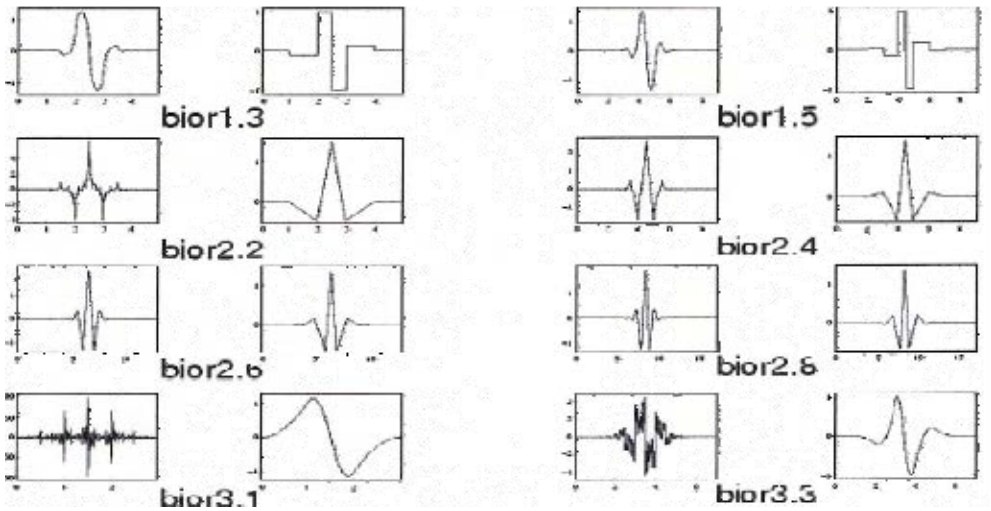


Figura 2.4.2. Apariencia de algunas waveles biortogonales.

Existen otras familias de *wavelets* ortogonales y biortogonales, como son:

- Wavelet Mexican Hat.
- Wavelet Coiflet.
- Wavelet Odergard.
- Wavelet Antonini., entre otras.

Una explicación detallada de esta transformada se proporciona en el siguiente capítulo.

2.5 Criterios de calidad objetiva

En varias aplicaciones se necesita el uso de cierto tipo de imágenes, por ejemplo imágenes de televisión, de rayos x, fotografías etc. Cada imagen posee sus propias características que las diferencian unas de otras, pero al mismo tiempo todas las imágenes tienen características promedio, en común.

Es necesario entonces contar con algún tipo de parámetro estadístico que permita evaluar la calidad de las imágenes, para así poder determinar que tanto se aproxima una imagen ya procesada a su imagen sin procesar (imagen original).

En la codificación de imágenes, las medidas promedio de calidad más usadas son el error cuadrático medio (MSE) y la relación señal a ruido (SNR), esto debido a que matemáticamente son fáciles de calcular.

Por otro lado, para aplicaciones multimedia y para compresión de imágenes, se ha incrementado el uso de medidas de calidad basadas en la percepción humana, dado a que es el usuario el que recibe el producto final de dichas aplicaciones. Entonces lo más apropiado es la calidad de una imagen que se encuentra basado en un modelo de percepción humana.

En este trabajo se tomarán como mediadas de calidad la razón señal pico a ruido “PSNR” y la “Función Costo de Entropía”, las cuales se explican a continuación.

2.5.1 Razón señal pico a ruido PSNR.

La razón señal a ruido es una medida promedio de la calidad de una imagen reconstruida comparada con una imagen original.

Como toda medida, se trata de obtener un valor promedio en el cual se pueda ver reflejada la calidad de una imagen reconstruida.

Entonces las imágenes reconstruidas con valores más altos se consideran como las mejores.

La razón señal pico a ruido PSNR es una medida de referencia con la que se trabajara, en donde se considera que la imagen patrón tiene un valor de tono de gris constante máximo 255 (para 8 bits por píxel), de aquí el nombre de señal pico, y cuya energía se compara con el error de reconstrucción cuadrático medio (MSE).

Entonces, se considera una imagen fuente *f(x,y)*, la cual contiene N por N píxeles y también se tiene una imagen reconstruida *F(x,y)*, el valor del error se lleva a cabo en la señal de luminancia, ya que los valores de los píxeles *f(x,y)* tienen un rango de tonos de gris entre el negro (0) y el blanco(255).

Primero se obtiene el error cuadrático medio (MSE) de la imagen reconstruida utilizando la expresión 2.5.1.

$$\text{MSE}= \sum \frac{[f(x,y)-F(x,y)]^2}{N^2} \qquad (2.5.1)$$

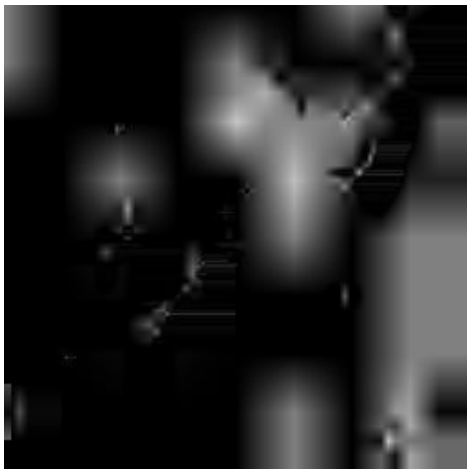
Luego, se compara ese error cuadrático medio con la energía de la imagen patrón de tono constante 255. Por lo que para obtener el PSNR se utiliza la expresión 2.5.2.

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \qquad (2.5.2)$$

El rango de los valores típicos de PSNR se encuentra entre 20[dB] y 40[dB], se utilizan generalmente dos decimales. El valor real no es significativo, pero la comparación entre dos valores para diversas imágenes reconstruidas, da una medida promedio de calidad.

En esta tesis se considerará que un valor PSNR mayor a 30[dB] es lo correcto.

A continuación se muestran algunas imágenes con su respectivo PSNR y el tamaño del archivo comprimido.



PSNR= 10.3511dB

Tamaño archivo comprimido 152 B

PSNR= 18.2772dB

Tamaño archivo comprimido 1.1KB

PSNR= 24.0295dB

Tamaño archivo comprimido 3.9KB

PSNR= 29.7665

Tamaño archivo comprimido 9KB

PSNR= 35.9176 dB
Tamaño archivo comprimido 14.4KB

PSNR= 41.3369
Tamaño archivo comprimido 31.9KB

PSNR= 49.2219dB
Tamaño archivo comprimido 50.4KB

Como se puede observar en las imágenes anteriores, entre mayor es el PSNR la calidad de la imagen es mejor, pero el archivo es más grande (ya que contiene mayor información).

2.5.2 Función costo de entropía

La entropía de una fuente de información es una medida promedio de la cantidad de información producida por la fuente. Definida por Shannon por:

$$H(X) = -\sum_{\forall i} p_i \log_2 p_i \qquad (2.5.3)$$

De acuerdo a esta definición, la función de costo de entropía esta definida por: $F(x) = -x \log x$. Su costo correspondiente es llamado entropía de la distribución de la energía dado por:

$$C(f) = -\sum_{m=1}^{N} \frac{\left|< f, g_m >\right|^2}{\| f \|^2} \log_e \frac{\left|< f, g_m >\right|^2}{\| f \|^2} \qquad (2.5.4)$$

Donde

$0 \leq C(f) \leq \log_e N$

$\left|< f, g_m >\right|^2$, es la energía de cada coeficiente.

$\| f \|^2$, se trata de la energía de la imagen original.

En la teoría de la información, el teorema de Shannon relativo a la codificación de una fuente sin ruido muestra que la cota inferior del número de bits para codificar individualmente cada producto interno $<f, g_m>$, es la entropía de la distribución de probabilidad de los valores de $<f, g_m>$. Esta distribución de probabilidad puede ser muy diferente de la distribución de las energías normalizadas utilizadas como medida de costo en esta tesis.

$$\frac{\left|< f, g_m >\right|^2}{\| f \|^2} \qquad (2.5.5)$$

Por ejemplo, si $<f, g_m> = \boldsymbol{A}$ para $0 \leq m < N$, entonces

$$\frac{|< f, g_m >|^2}{\|f\|^2} = \frac{1}{N} \qquad (2.5.6)$$

y **el costo** $C(f, B) = \log_e N$ **es máximo**. Mientras que la distribución de probabilidad del producto interno es una función de Dirac discreta localizada en A y su entropía es mínima e igual a cero.

En seguida se muestran algunas imágenes con distinto costo de entropía, así como su respectivo PSNR.

Nivel 1.
Costo =2.77157.
PSNR =26.5996dB.

Nivel 5
Costo =0.198169
PSNR =24.0295dB

Como se puede observar en las imágenes anteriores entre mayor costo de entropía se tenga mayor es la calidad de la imagen.

Existen otras medidas de calidad, entre ellas se pueden mencionar:

- Distancia de Hellinger.
- Fidelidad de la imagen.
- Error espectral de fase.
- Medida tasa de distorsiòn.
- Norma HV SL2.
- Correlación Czekonowski.

CAPÍTULO III Análisis multiresolución y la transformada wavelet

En este capítulo se explica el concepto de análisis multiresolución, que es una herramienta muy útil para analizar una señal, ya que con este análisis se puede representar la señal con varias resoluciones, lo cual se aprovecha para la transmisión progresiva.

A continuación se da una explicación breve de las funciones *wavelet* y escala con las cuales se realiza la transformación. Partiendo de la naturaleza de estas funciones, se describe el proceso de transformación *wavelet* pero de manera discreta presentando sus ventajas sobre la continua, y posteriormente la aplicación de esta transformada a señales de una y dos dimensiones.

Ya que para aplicar la DWT es necesario un método de filtrado, se da una descripción del método que se utilizó llamado filtrado *lifting* o por desplazamiento.

Debido a la utilidad de la transformada *Wavelet*, en la parte última del capítulo se mencionan algunas otras aplicaciones en las que ésta se utiliza, con la finalidad de tener una idea de los campos en los que también se puede usar como herramienta.

3.1 Análisis multiresolución basado en la transformada wavelet

El análisis multiresolución (MRA) constituye la etapa más importante en el uso de funciones escala y *wavelets,* así como en el desarrollo de algoritmos. Como el nombre lo indica, en el análisis multiresolución (MRA) se estudia una función a varios niveles de aproximación, o resolución. Esta idea fue desarrollada por Meyer [6] y Mallat [7].

Con este tipo de análisis, se puede separar una función complicada en varias más simples y estudiarlas de manera independiente. Tomemos el ejemplo de la figura 3.1.1.

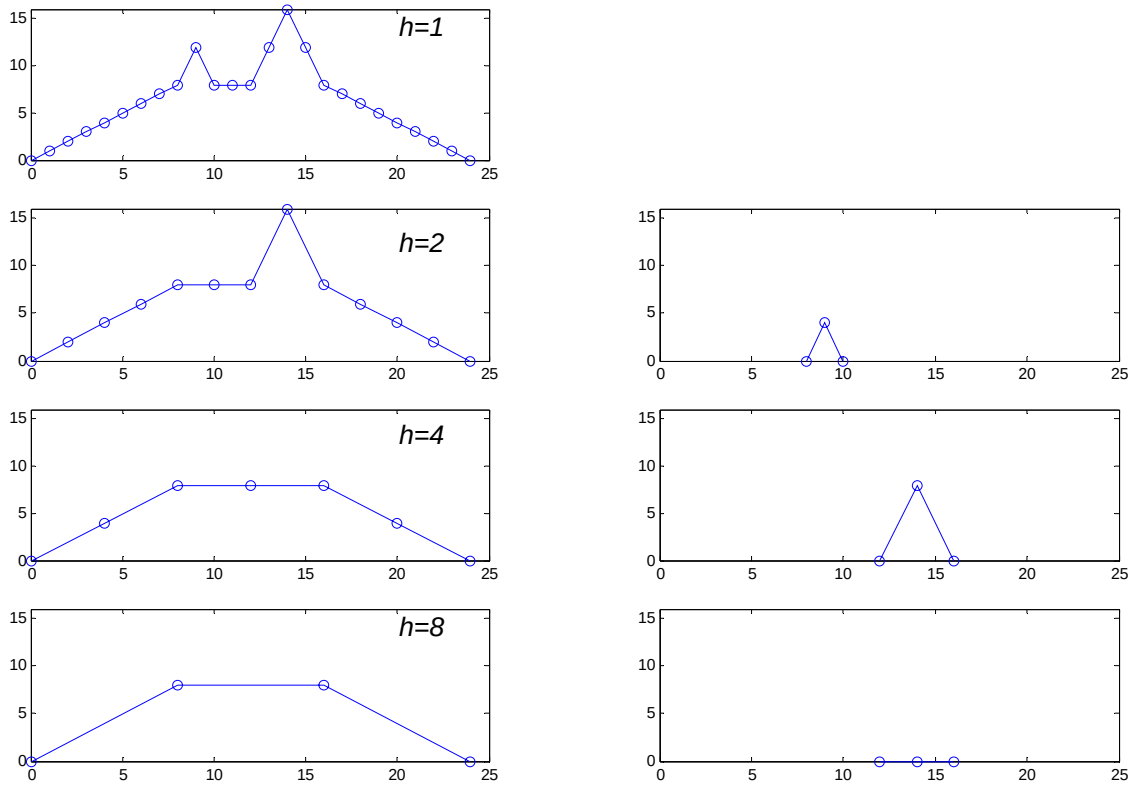


Figura 3.1.1 Representación multinivel de una función

Si se quiere representar esta función con un solo nivel de aproximación, se necesita discretizarla usando pasos de tamaño h. Utilizando varios pasos de discretización (o resoluciones) para representar la función, se pueden reducir significativamente el número de puntos requeridos para su representación

exacta. De esta manera, la función original es representada con una aproximación burda, junto con sus detalles de cada nivel de resolución. Se puede observar que con cada nivel (o escala), el tamaño h se va duplicando. Por analogía con la música donde una octava es el intervalo de dos sonidos cuyas frecuencias fundamentales tienen una relación de dos a uno, cuando la escala h se va duplicando se dice que se tiene una representación de la señal a nivel de octavas.

En la figura 3.1.1 se observa que cada vez que bajamos de nivel duplicando el intervalo de muestreo (h), se extraen porciones de la función, mostradas a la derecha de la figura. Por lo tanto, la parte sobrante (el lado izquierdo) será nuevamente descompuesta en los niveles sucesivos.

Para formalizar este análisis, en la figura 3.1.1 podemos asignar a todas las funciones del lado izquierdo la letra A_s, y aquellas que están del lado derecho la letra W_s, donde s representa el número de la escala. Desde el punto de vista del análisis de señales, podemos decir que A_s es generada por la base de funciones:

$$\phi_{k,s} : 2^{\frac{s}{2}} \phi(2^s t - k); \qquad k \in Z$$

Y W_s por la base de funciones:

$$\psi_{k,s} : 2^{\frac{s}{2}} \psi(2^s t - k); k \in Z$$

Cualquier par de funciones $x_s(t)$ e $y_s(t)$ puede ser representado como una combinación lineal de las bases $\phi_{k,s}(t)$ y $\psi_{k,s}(t)$ respectivamente. Se puede observar en el análisis multiresolución que las funciones $x_{s-1}(t) \in A_{s-1}$ e $y_{s-1}(t) \in W_{s-1}$ se derivan de $x_s \in A_s$, función del nivel superior, por lo tanto, debemos esperar que las dos bases $\phi_{k,s-1}(t)$ de A_{s-1} y $\psi_{k,s-1}(t)$ de W_{s-1} están

relacionadas de alguna manera con la base del nivel superior $\phi_{k,s}(t)$ de A_s. Esta relación nos ayudará a concebir un algoritmo que permita obtener las funciones x_{s-1} y y_{s-1} a partir de x_s más eficientemente.

Para alcanzar un análisis multiresolución de una función como la mostrada en la figura 3.1.1, necesitamos una función de energía finita $\phi(t) \in L^2(R)$, llamada *función de escalamiento*, la cual genera una secuencia anidada $\{A_j\}$:

$$\{0\} \leftarrow ... \subset A_{-1} \subset A_0 \subset A_1 \subset ... \rightarrow L^2,$$

y satisface la ecuación de dilatación (refinamiento):

$$\phi(t) = \sum_k g_0[k]\phi(at-k) \qquad (3.1.1)$$

para alguna $a > 0$ y coeficientes $\{g_0[k]\} \in \ell^2$.

Se considera $a = 2$, lo cual corresponde a escalas similares a las octavas. La función $\phi(t)$ es representada como la superposición de una versión de sí misma pero *escalada* y trasladada (de ahí el nombre de función de *escalamiento*). Esto quiere decir que todos los espacios son versiones escaladas del espacio origen A_0 [8] el cual es generado por $\{\phi(\cdot - k) : k \in Z\}$ y de forma más general, A_s por $\{\phi_{k,s} : k, s \in Z\}$ teniendo como resultado las siguientes propiedades:

$$x(t) \in A_s \Leftrightarrow x(2t) \in A_{s+1} \qquad (3.1.2)$$

$$x(t) \in A_s \Leftrightarrow x(t + 2^{-s}) \in A_s \qquad (3.1.3)$$

Existen muchas funciones que generan secuencias de subespacios anidados. Pero la ecuación de refinamiento 3.1.1 y las propiedades 3.1.2 y 3.1.3 son únicas para un Análisis Multiresolusión.

Como A_s es un subespacio propio de A_{s+1} para cada s, existe un espacio sobrante en A_{s+1}, llamado W_s, el cual, combinado con A_s, resulta A_{s+1}. Este espacio $\{W_s\}$ es llamado el *subespacio wavelet* y es complementario a A_s en A_{s+1}. Esto quiere decir que:

$$A_s \cap W_s = \{0\}, \qquad s \in Z$$

$$A_s \oplus W_s = A_{s+1}, \qquad s \in Z$$

El símbolo $\oplus$ representa la suma de complementos ortogonales.

Los subespacios $\{W_s\}$ son generados por la función $\psi(t) \in L^2$, llamada función *wavelet*, así como $\{A_s\}$ es generado por $\phi(t)$. En otras palabras, cualquier $x_s(t) \in A_s$ puede ser expresada como:

$$x_s(t) = \sum_k a_{k,s}\phi(2^2 t - k),$$

Y cualquier función $y_s(t) \in W_s$ puede ser expresada como:

$$y_s(t) = \sum_k w_{k,s}\phi(2^2 t - k)$$

donde los coeficientes $\{a_{k,s}\}k \in Z$, $\{w_{k,s}\}k \in Z \in \ell^2$

Como:

$$A_{s+1} = W_s \oplus A_s$$

$$= W_s \oplus W_{s-1} \oplus A_{s-1}$$

$$= W_s \oplus W_{s-1} \oplus W_{s-2} \oplus A_{s-2} \ldots,$$

Entonces tenemos:

$$A_s = \bigoplus_{\ell=-\infty}^{s-1} W_\ell$$

Una representación esquemática de la naturaleza jerárquica de A_s y W_s es mostrada en la siguiente figura:

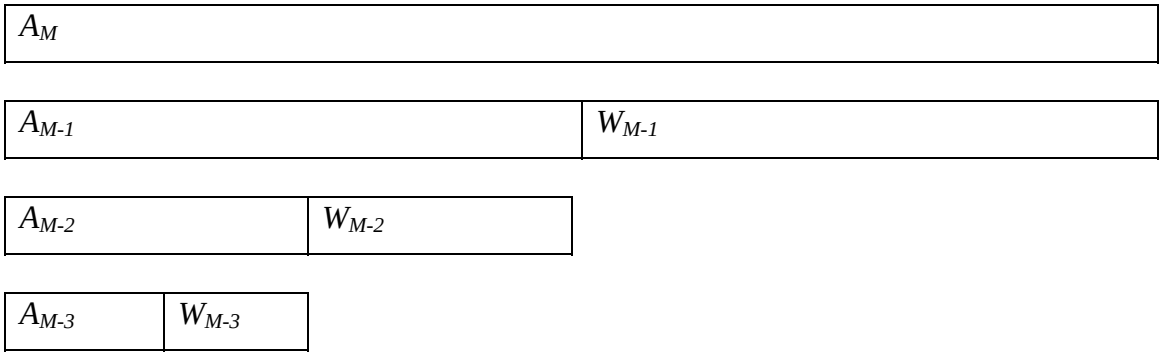


Figura 3.1.2 Separación de subespacios en un análisis multiresolución MRA

Se puede observar que los subespacios $\{A_s\}$ están anidados, mientras que $\{W_s\}$ son mutuamente ortogonales. Consecuentemente, tenemos:

$$\begin{cases} A_\ell \cap A_m = A_\ell , & m > \ell \\ W_\ell \cap W_m = \{0\} & m \neq \ell \\ A_\ell \cap W_m = \{0\} , & \ell \leq m \end{cases}$$

3.2 La función wavelet.

Las *wavelets* son una familia de funciones generadas a partir de una función inicial Ψ mediante operaciones de dilatación y translación de la misma. La función *wavelet* está definida por:

$$\Psi^{a,b}(t)=|a|^{-1/2}\,\Psi\left(\frac{t-b}{a}\right)$$

donde t es una variable y Ψ es la función *wavelet* llamada madre. La idea básica de la transformada *wavelet* es representar cualquier función arbitraria f como una superposición de funciones *wavelets*. Tal superposición descompone a la función f en niveles de diferente escala, a su vez cada nivel es descompuesto de nuevo con una resolución adaptada a ese nivel. Si se inicia la discretización con $a=a_0^m$ y $b=nb_0^m a_0^m$ siendo m y n enteros y $a_0>1$ y $b_0>0$ ambos fijos, entonces puede ser escrita la función f como una sumatoria discreta de superposiciones. Luego, la descomposición discreta de *wavelet* viene dada por:

$$f=\sum C_{m,n}(f)\Psi_{m,n}$$

donde:

$$\Psi_{m,n}(t)=\Psi^{a_o^m,nb_0a_0^m(t)}=a_0^{-m/2}\,\Psi(a_0^{-m}t-nb_0)$$

3.3 La función de escala

Desde el punto de vista de una descomposición de un espectro en varias sub-bandas, una *wavelet* se puede ver como un filtro paso banda, entonces una serie de *wavelets* ensanchadas puede ser considerada como un banco de filtros paso banda con Q (relación entre la frecuencia central y el ancho de banda) constante para todos.

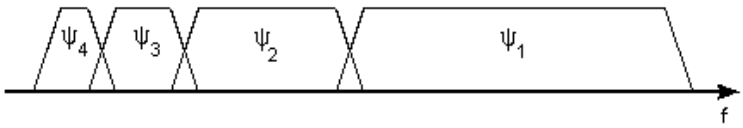


Figura 3.3.1 Banco de filtros paso banda con Q constante

El espectro de la señal se puede cubrir en su totalidad estrechando cada vez la *wavelet* en el dominio del tiempo con un factor de 2. Es decir que se cubre primero la mitad del espectro de la señal con una *wavelet*, después se contrae la *wavelet* y se cubre la mitad de la parte restante del espectro y así sucesivamente, dando origen a lo que se conoce como filtrado de media banda. Esto implicaría un número infinito de etapas de filtrado para poder cubrir todo el espectro de la señal analizada, sin embargo, se puede usar una función paso bajas (como un corcho) para cubrir la parte restante del espectro una vez que ésta sea lo suficientemente pequeña. Esta función es llamada **función de escalamiento**.

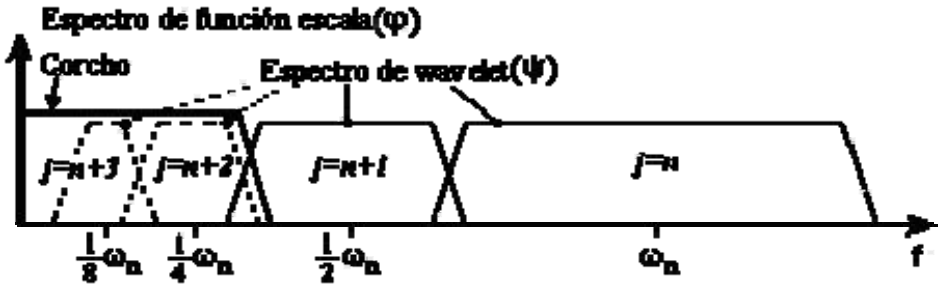


Figura 3.3.2 La función escala cubre el espectro restante

Dado lo anterior podemos deducir que la representación tiempo-escala de una señal digital se obtiene usando técnicas de filtrado digital.

3.4 La transformada wavelet discreta, DWT, en 2 dimensiones

La transformada *wavelet* discreta (DWT) puede ser trasladada y escalada por pasos discretos y no continuos como la CWT.

$$\Psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \Psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j} \right)$$

El efecto de discretizar la *wavelet* es que el espacio tiempo-escala es muestreado en intervalos discretos s_0, donde generalmente $s_0=2$ de manera que corresponda a una escala diádica, y el factor de traslación $\tau_0=1$, para que también se tenga un muestreo diádico en el eje del tiempo.

En la siguiente figura se muestra la localización de las *wavelets* discretas en el plano tiempo-escala en una rejilla diádica, donde se observa que en las escalas pequeñas se tienen más muestras en el tiempo. Al aumentar la escala, en cada paso se reduce el número de muestras a la mitad.

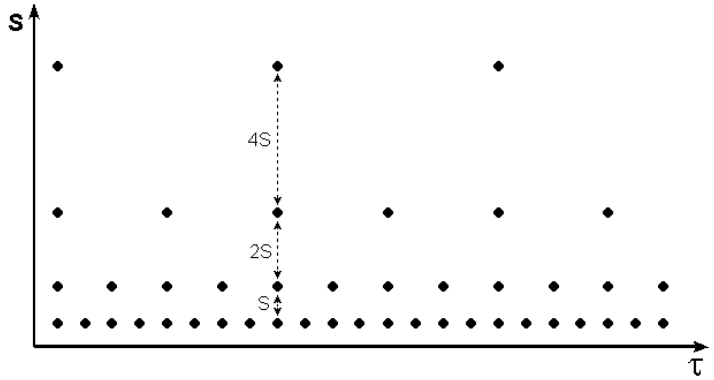


Figura 3.4 Localización de las *wavelets* discretas en el plano tiempo-escala

3.4.1 Descomposición de una señal

El proceso de descomposición comienza pasando la señal discretizada a través de un filtro paso bajas de media banda con respuesta al impulso $h[n]$. En el dominio del tiempo, el filtrado de la señal corresponde a la operación matemática de **convolucionar** ésta con la respuesta al impulso del filtro $h[n]$. Este filtro pasa bajas de media banda elimina las componentes frecuenciales situadas por encima de la mitad del ancho de banda de la señal.

La *resolución*, que es una medida de la cantidad de detalle, varía por la operación de filtrado, mientras que la *escala* varía mediante operaciones de submuestreo, el cual es posible debido al filtrado de media banda previo, y que consiste en reducir la tasa de muestreo eliminando la mitad de las muestras de la señal en cada paso.

La transformada discreta *wavelet* (DWT) analiza la señal a diferentes bandas de frecuencias con diferentes resoluciones, descomponiendo la señal en cada paso en una aproximación y en un detalle. Con este propósito se emplean los dos conjuntos de funciones mencionados anteriormente denominadas **funciones de escalamiento** y **funciones *wavelets***, las cuales están asociadas a filtros de media banda **paso bajas** y **paso altas** respectivamente. La descomposición de la señal en diferentes bandas de frecuencia se obtiene al pasar la señal original $x[n]$ sucesivamente por un filtro paso bajas $h[n]$ y paso altas de media banda $g[n]$. Después tomamos $h[n]$, y al haberse reducido su frecuencia máxima a la mitad, se puede eliminar la mitad de las muestras de acuerdo al teorema de Nyquist (la frecuencia de muestreo debe ser mayor o igual al doble de la frecuencia máxima de la señal muestreada: $f_{muestreo} \geq 2f_{máx}$).

De esta manera se ha constituido el primer nivel de descomposición, lo que matemáticamente puede expresarse como:

$$y_{high}[k] = \sum_n x[n] \cdot g[2k - n]$$
$$y_{low}[k] = \sum_n x[n] \cdot h[2k - n] \qquad (3.4.1)$$

Donde $y_{high}[k]$ y $y_{low}[k]$ son las salidas de los filtros paso altas y paso bajas respectivamente, después del submuestreo de orden 2. La salida de los filtros pasa altas y pasa bajas también pueden expresarse como $d(n)$ y $c(n)$ respectivamente, donde $d(n)$ son los detalles y $c(n)$ es la aproximación. El procedimiento anterior también esta clasificado como codificación en subbandas y puede repetirse para conseguir un mayor orden de descomposición.

Con este procedimiento, en cada etapa, el filtrado dará como resultado una disminución a la mitad del ancho de banda (resolución en frecuencia duplicada), y el submuestreo resultará en una disminución a la mitad del número de muestras (resolución en el tiempo dividida). En esto se observa el principio de incertidumbre de Heisenberg, es decir, mientras la resolución en la frecuencia aumenta, la resolución en el tiempo disminuye. La salida del filtro pasa altas tendrá la mitad de muestras que la señal original, las cuales constituyen el primer nivel de los coeficientes de la DWT. La salida del filtro pasa bajas también tendrá la mitad de muestras y esta señal se descompondrá nuevamente en el siguiente nivel pasándola a su vez por filtros paso altas y paso bajas, obteniendo el segundo nivel de los coeficientes de la DWT.

El proceso continúa hasta que el número de muestras obtenidas por los filtros ya no sean divisibles entre 2. La DWT de la señal original se obtiene entonces concatenando todos los coeficientes, comenzando desde el último nivel de descomposición.

La DWT tendrá por tanto el mismo número de coeficientes que la señal original, es decir, no se genera redundancia durante el proceso.

A continuación se muestra gráficamente un ejemplo: H_1 es el filtro pasa altas correspondiente a la función *wavelet* dando como resultado los coeficientes de la DWT en cada etapa, H_0 es el filtro pasa bajas, correspondiente a la función de escalamiento y dando como resultado la aproximación. Los bloques ↓2 corresponden al submuestreo de orden 2.

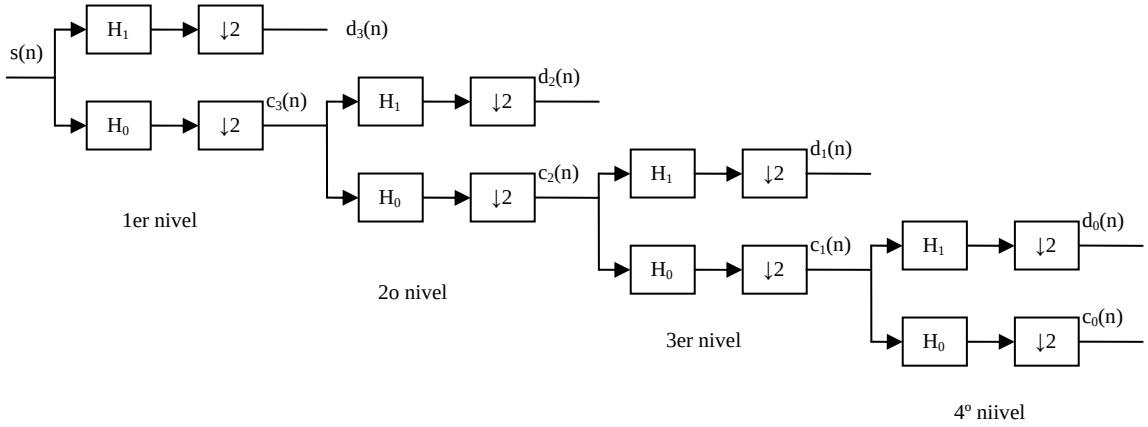


Figura 3.4.1 Descomposición de la señal s(n) en 4 niveles

Si $s(n)$ fuera una función de 16 muestras, la transformada *wavelet* contendría el mismo número de términos, los cuales consisten en los coeficientes de detalles: d_3 (2^3=8 muestras), d_2 (2^2=4 muestras), d_1 (2 muestras), d_0 y la aproximación c_0 (1 muestra cada uno).

La relación entre las respuestas impulso de los filtros pasa altas y pasa bajas es a través de la siguiente ecuación:

$$g[L\text{-}1\text{-}n]=(-1)^n \cdot h[n] \qquad (3.4.2)$$

donde $g[n]$ es el filtro pasa altas, $h[n]$ es el filtro pasa bajas y L es la longitud del filtro expresada en número de coeficientes. La conversión de paso bajas a paso altas se hace a través del factor $(-1)^n$, los filtros que satisfacen esta característica se conocen como Filtros Espejos en Cuadratura (QMF por sus siglas en inglés). Las dos operaciones de filtrado y submuestreo se pueden expresar como:

$$(Gf)_k = y_{high}[k] = \sum_n x[n] \cdot g[-n + 2k] \qquad (3.4.3)$$

$$(Hf)_k = y_{low}[k] = \sum_n x[n] \cdot h[-n + 2k] \qquad (3.4.4)$$

3.4.2 Reconstrucción de la señal

La reconstrucción en este caso es muy simple cuando los filtros de media banda forman una base **ortonormal**. Para estos efectos el procedimiento anterior se sigue en sentido inverso de modo que la señal es interpolada de orden 2 insertando ceros entre cada muestra y pasada a través de los filtros de síntesis $g'[n]$ y $h'[n]$, pasa altas y pasa bajas respectivamente, para posteriormente sumarse ambas salidas. Con este tipo de *wavelets* ortogonales los filtros de análisis y síntesis, respectivos, son idénticos. Cuando las *wavelets* son biortonormales, son necesarios diferentes filtros para la descomposición y para la reconstrucción.

3.4.3 Transformada *wavelet* en dos dimensiones 2D

Para *wavelets* separables de dos dimensiones se considera a la señal de entrada como una matriz cuadrada (N x N), se descompone en la dirección de los renglones y a cada uno se le aplica la transformada *wavelet* de una dimensión. Para completar la transformación 2D al resultado de esta etapa de descomposición se le aplica luego la descomposición en la dirección de las columnas.

Primero se procesan los renglones de la señal aplicando a cada renglón la transformada *wavelet* 1D. Entonces se obtienen dos matrices rectangulares de tamaño N x N/2, una resultante del filtro paso bajas y otra del paso altas. Después, se procesan las columnas de las matrices resultantes para obtener

cuatro matrices de N/2 x N/2 llamadas: (LL_1, LH_1, HL_1, HH_1), tal y como se muestra en la figura 3.4.2.

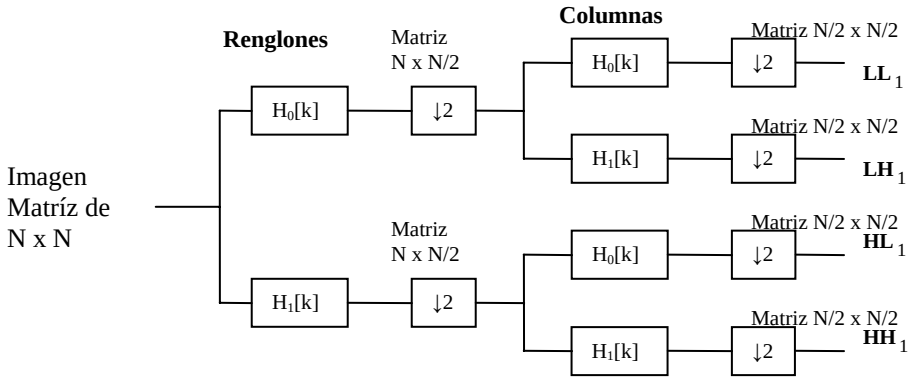


Figura 3.4.2 Descomposición de una señal de dos dimensiones en un nivel

Con esto se termina un nivel de descomposición. Para el siguiente nivel se toma la matriz resultante del filtrado paso bajas tanto en la dirección de los renglones como en la dirección de las columnas (LL_1) y se realiza el mismo procedimiento.

La DWT de la señal original para el primer nivel de descomposición se obtiene concatenando los coeficientes, siguiendo la distribución que se muestra en la figura 3.4.3.

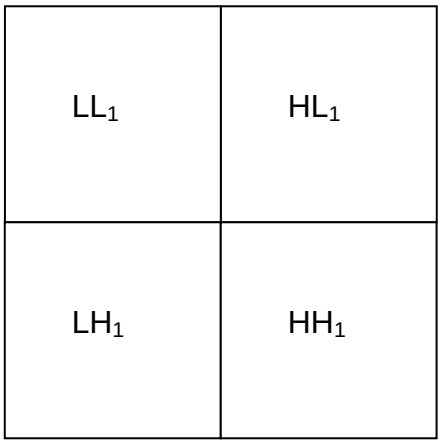


Figura 3.4.3 Distribución de las matrices resultantes de un nivel de descomposición

Para obtener la transformada completa se concatenan todos los coeficientes de todos los niveles siguiendo la distribución anterior, comenzando desde el último nivel de descomposición.

Por ejemplo si se tuviera una imagen de 4 x 4, entonces se tendrían 2 niveles de descomposición posibles (2 y 1) y sus coeficientes ordenados siguiendo la distribución que se mostró anteriormente quedarían como se muestra en la figura 3.4.4 donde la matriz LL_2 a su vez fue descompuesta en LL_1,HL_1,LH_1 y HH_1.

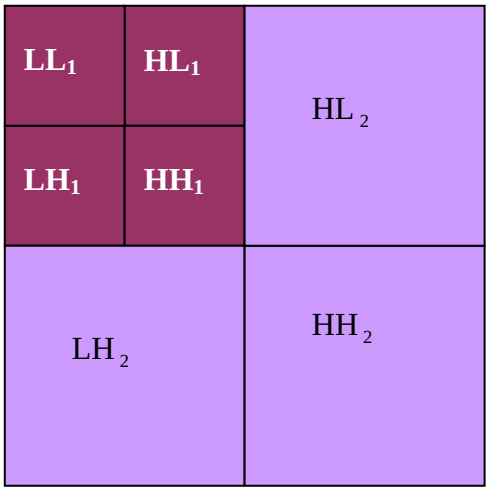


Figura 3.4.4 Distribución de una imagen transformada con 2 niveles de descomposición

Sin embargo la manera de filtrar por medio del uso de la convolución lineal no es adecuada ya que el número de coeficientes resultantes excede el número de coeficientes originales, o existe un desplazamiento de la señal. Existe una técnica que es fácil de realizar computacionalmente y resuelve el problema del número de coeficientes, esta consiste en el filtrado por desplazamiento (lifting).

3.5 Filtrado por desplazamiento (*Lifting*)

Como se ha mencionado antes, podemos llevar a cabo la transformada *wavelet,* o codificación subbanda, o multiresolución, usando un banco de filtros. Una etapa simple de un banco de filtros es mostrado en la figura 3.5.1, y puede ser realizado usando filtros FIR (de Respuesta Finita al Impulso).

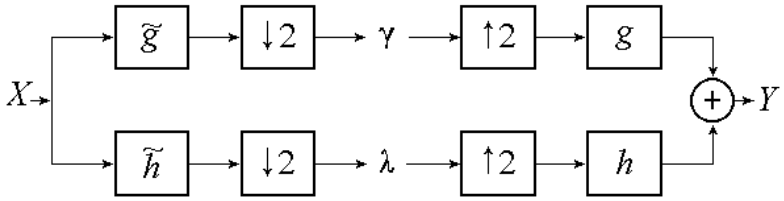


Figura 3.5.1: Una etapa de un banco de filtros de análisis y reconstrucción.

En la figura se observa los filtros de análisis ($\tilde{h}$ y $\tilde{g}$), y los de síntesis (h y g) seguidos de un submuestreo de orden 2 para la transformada directa y un sobremuestre de orden 2 para la transformada inversa.

Según la figura, se realiza la operación de filtrado y luego se desechan la mitad de las muestras al submuestrear. Se puede observar que este procedimiento no es eficiente, ya que hay una pérdida de tiempo al calcular muestras que se van a eliminar, por lo que sería más conveniente submuestrear la señal al mismo tiempo de filtrarla para ahorrar algunos cálculos.

De igual forma ocurre con la síntesis, ya que antes de filtrar la señal se sobremuestrea, lo cual no es otra cosa que insertar ceros entre cada dos muestras, esto implica muchas multiplicaciones por cero al filtrar, lo cual también es una pérdida de tiempo.

Analizando el diagrama de un filtro FIR con el submuestreo en la salida, se puede obtener un diagrama más eficiente, el cual se muestra en la figura 3.5.2.

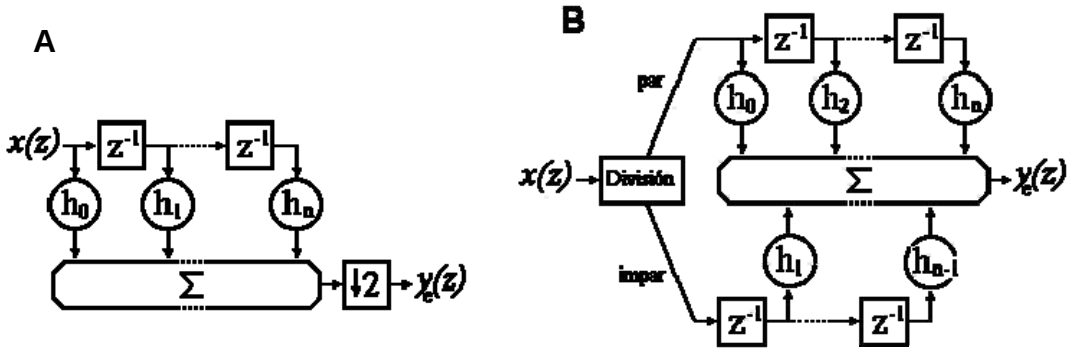


Figura 3.5.2 A) Filtro FIR estándar con submuestreo a la salida. B) una implementación más eficiente.

Analizando los resultados a la salida del primer diagrama, se puede llegar a un método para realizar el filtrado de manera más eficiente tanto para la transformada como la antitransformada.

El filtrado por desplazamiento resuelve de manera eficiente el problema, ya que consiste en una secuencia de operaciones muy simples, mediante las cuales las muestras impares de la señal se actualizan con una suma ponderada de las muestras pares, y las muestras pares se actualizan con una suma ponderada de las muestras impares, alternadamente [21].

3.5.1 Transformada *Wavelet* Directa

La figura 3.5.3 muestra el diagrama de bloques que representa la transformada *wavelet* usando el filtrado por desplazamiento.

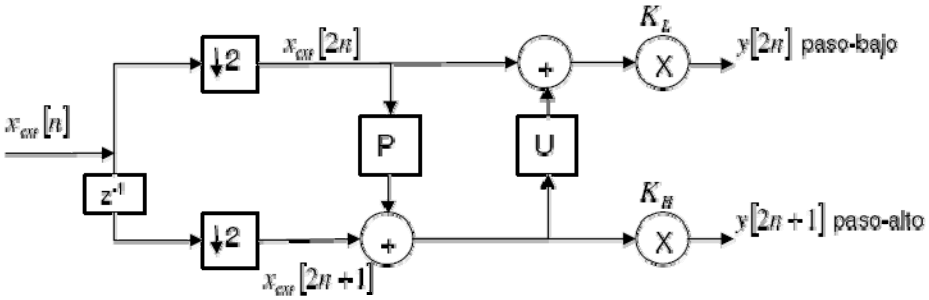


Figura 3.5.3 Transformada *Wavelet* directa usando filtrado por desplazamiento

Donde x_{ext} es la función de la señal original pero extendida. P y U son las reglas de predicción y actualización respectivamente, y K_L y K_H son constantes de escalado, que dependerán de los filtros *wavelet* empleados. Las muestras pares de la señal de salida $y[n]$ corresponden a las muestras paso-bajo ($y[2n]$), mientras que las muestras impares corresponden a las muestras paso-alto ($y[2n+1]$).

Para comenzar con el proceso de filtrado, se debe extender la señal, esto para poder realizar el filtrado en los bordes de la señal. Si no se extendiera la señal, se estarían agregando muchas frecuencias muy grandes, ya que al inicio y al final de la señal habría un cambio drástico entre el último valor y cero.

La señal original se extiende añadiendo un cierto número de muestras a la izquierda y a la derecha de la señal dependiendo el filtro que se vaya a utilizar. Para la realización de este trabajo se emplearon dos filtros: el Daubechies 9/7 y el LeGall 5/3 que son los que se emplean en la norma JPEG 2000.

El tipo de extensión que se le aplica a la señal es de tipo simétrica y periódica. Una extensión simétrica consiste en extender la señal con las muestras obtenidas mediante la reflexión de la señal tanto para la izquierda como para la derecha. Es periódica para los casos en los que el número de muestras no sea suficiente para extender la señal, principalmente para los niveles más bajos de descomposición.

Suponiendo una señal de tamaño N, la primera muestra de la señal tiene la posición i_0 y la última tiene la posición i_{N-1}. La tabla 3.1 y 3.2 muestra la cantidad de elementos que hay que agregar a la señal a la derecha y a la izquierda respectivamente dependiendo de la posición de la primera y última muestra, tanto para el filtro LeGall 5/3 como para Daubechies 9/7.

i_0	Legall 5/3	Daub (9/7)
Par	2	4
Impar	1	3

Tabla 3.1 Extensión a la izquierda para la transformada directa

i_N	Legall 5/3	Daub (9/7)
Par	1	3
Impar	2	4

Tabla 3.2 Extensión a la derecha para la transformada directa

Transformada *Wavelet* usando el filtro Legall 5/3

La transformada *wavelet* con el filtro Legall 5/3 es reversible, ya que la señal original y la resultante son valores enteros. Por lo tanto, sólo debe aplicarse a señales cuyos valores son enteros.

La transformada directa con este filtro consiste en dos pasos de "desplazamiento" [16]:

$$\begin{aligned} y[2n+1] &= x_{ext}[2n+1] - \left\lfloor \frac{x_{ext}[2n] + x_{ext}[2n+2]}{2} \right\rfloor \qquad \left\lceil \frac{i_0}{2} \right\rceil - 1 \le n < \left\lceil \frac{i_N}{2} \right\rceil \\ y[2n] &= x_{ext}[2n] - \left\lfloor \frac{y[2n-1] + y[2n+1]}{4} \right\rfloor \qquad \left\lceil \frac{i_0}{2} \right\rceil - 1 \le n < \left\lceil \frac{i_N}{2} \right\rceil \end{aligned} \qquad \textbf{(3.5.1)}$$

donde x_{ext} es la señal extendida siguiendo la explicaión anterior.

Estos pasos predicen y actualizan respectivamente las muestras pares e impares en la señal de salida $y[n]$. Los corchetes $\lfloor \cdot \rfloor$ indican una operación de redondeo inferior y los corchetes $\lceil \cdot \rceil$ indican redondeo superior, de manera que se asegura que a la salida, todos los valores serán enteros (recordemos que los valores de la señal de entrada deben ser también enteros).

Los elementos i_0 e i_{N-1} representan las posiciones de la primer y última muestra de la señal $x[n]$ respectivamente. La señal de salida $y[n]$ está compuesta por las muestras resultantes del filtro paso bajas y paso altas de forma alternada. Si i_0 es impar, la primera muestra de $y[n]$ es una muestra paso-alto. Si i_0 es par, la primera muestra de $y[n]$ es una muestra paso-bajo.

En una señal de 8 muestras, por ejemplo, la primera muestra se encuentra en la posición 0 (i_0=0) y la última en la posición 7 (i_{N-1}=7 y por lo tanto i_N=8).

De acuerdo a la tabla 3.1 y 3.2, se agregan 2 elementos a la izquierda y 1 a la derecha. Las muestras serán reflejadas tomando como centro la primer y última muestra de la señal.

$$x[n] = \text{A B C D E F G H}$$

$$x_{ext}[n] = \textbf{C B} \text{ A B C D E F G H } \textbf{G}$$

Ahora se pueden aplicar los dos pasos de la transformación. La figura 3.6.1 muestra en forma de diagrama la implementación.

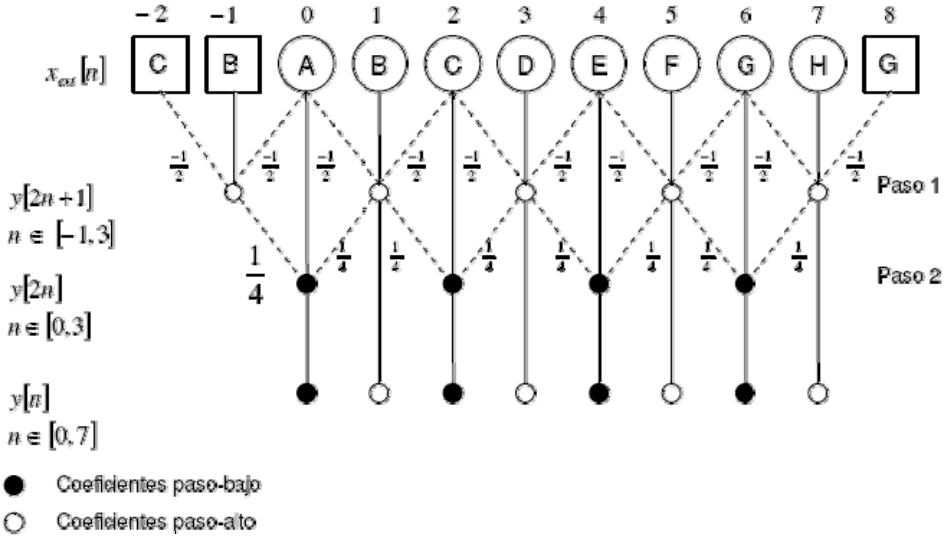


Figura 3.5.4 Diagrama de la DWT con el filtro Legall 5/3 para una señal de 8 muestras

Todas las líneas indican una multiplicación de la muestra por el número que tienen a un lado (- ½ ó ¼), si no tienen número, quiere decir que se multiplican por 1. Las líneas punteadas indican el redondeo especial. Los puntos donde se encuentran las líneas equivalen a un sumador.

Para el caso de una imagen, se puede aplicar esta transformación a cada renglón y separar las muestras para tener primero las paso bajas y después las paso altas, Después se aplica el procedimiento a las columnas y se vuelven a ordenar las muestras resultantes para tener las matrices en el orden necesario.

Transformada *Wavelet* usando el filtro Daubechies 9/7

La DWT con el filtro Daubechies 9/7 es irreversible, ya que el resultado no son elementos enteros. Esta transformada consiste en 4 pasos de desplazamiento y 2 pasos más de escalado:

$$
\begin{aligned}
&paso1: y[2n+1] = x_{ext}[2n+1] + \left(\alpha[x_{ext}[2n] + x_{ext}[2n+2]]\right)\\
&paso2: y[2n] = x_{ext}[2n] + \left(\beta[y[2n-1] + y[2n+1]]\right)\\
&paso3: y[2n+1] = y[2n+1] + \left(\gamma[y[2n] + y[2n+2]]\right)\\
&paso4: y[2n] = y[2n] + \left(\delta[y[2n-1] + y[2n+1]]\right)\\
&paso5: y[2n+1] = -ky[2n+1]\\
&paso6: y[2n] = \left(1/k\right)y[2n]
\end{aligned}
$$

$$
\begin{aligned}
&i_0 - 3 \le 2n+1 < i_N + 3\\
&i_0 - 2 \le 2n < i_N + 1\\
&i_0 - 1 \le 2n+1 < i_N + 1\\
&i_0 \le 2n < i_N\\
&i_0 \le 2n+1 < i_N\\
&i_0 \le 2n < i_N
\end{aligned}
\qquad \textbf{Ecuación (3.5.2)}
$$

donde:

$$
\begin{cases}
\alpha = -1.586134342\\
\beta = -0.052980118\\
\gamma = 0.882911075\\
\delta = 0.443506852
\end{cases}
$$

y $k = 1.230174105$

Siguiendo con el ejemplo de la señal de 8 muestras, sabemos que i_0 e i_N son pares, por lo tanto, de acuerdo la tabla 3.1 y 3.2, se deben agregar 4 elementos a la izquierda y 3 a la derecha reflejando las muestras tomando como centro la primer y última muestra.

$$x[n] = \text{A B C D E F G H}$$
$$x_{ext}[n] = \textbf{\textit{E D C B}} \text{ A B C D E F G H } \textbf{\textit{G F E}}$$

Ahora se aplican los 6 pasos de la transformación como en la figura 3.5.5.

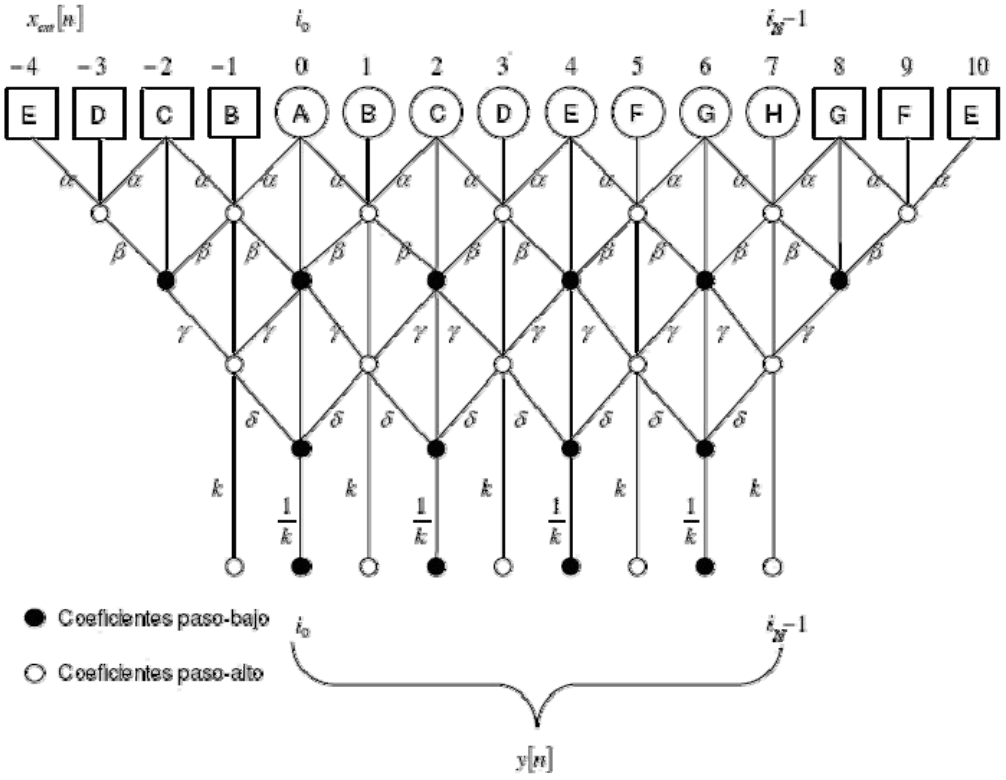


Figura 3.5.5 Diagrama de la DWT con el filtro Daubechies 9/7 para una señal de 8 muestras

Todas las líneas indican la multiplicación de una muestra por la constante que tiene a un lado, si la línea no tiene un número a un lado, quiere decir que se

multiplica por uno. Los puntos donde confluyen las líneas equivalen a un sumador.

Con este paso se concluye un nivel de descomposición en la transformada *Wavelet*. Para el caso de las imágenes, se debe aplicar este procedimiento a cada renglón, del resultado se agrupan los elementos paso bajas y paso altas para aplicarles el mismo proceso a las columnas y agrupar nuevamente los elementos paso bajas y paso altas de manera que se obtengan las matrices en el orden necesario.

3.5.2 Transformada *Wavelet* Inversa

Para el caso de la Transformada *Wavelet* Discreta Inversa se tiene el esquema de la figura 3.5.6:

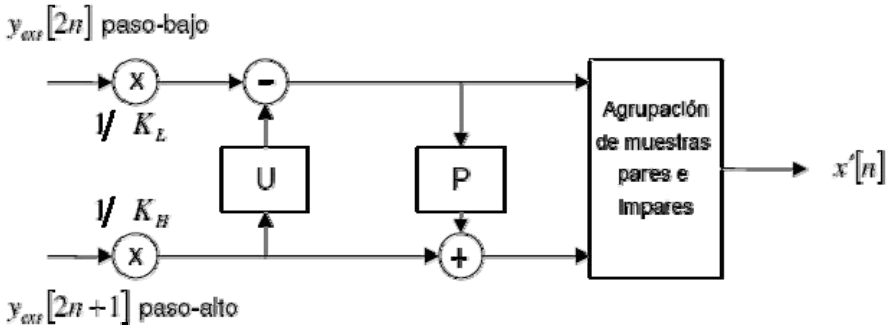


Figura 3.5.6 Transformada *Wavelet* inversa usando filtrado por desplazamiento

donde P y U son las reglas de predicción y actualización respectivamente, y K_L y K_H son constantes de escalado, que dependerán de los filtros *wavelet* empleados.

El número de elementos que se agregan a la derecha para la transformada directa, se agregan a la izquierda para la transformada inversa, al igual que el número de elementos que se agregan a la izquierda en la transformada directa se agregan a la derecha en la inversa.

Transformada *Wavelet* Inversa usando el filtro Legall 5/3

La Transformada *Wavelet* Inversa con este filtro también se compone de dos pasos:

$$\begin{aligned} x[2n] &= y_{ext}[2n] - \left\lfloor \frac{y_{ext}[2n-1] + y_{ext}[2n+1] + 2}{4} \right\rfloor \qquad i_0 - 1 \le 2n < i_N - 1 \\ x[2n+1] &= y_{ext}[2n+1] + \left\lfloor \frac{x[2n] + x[2n+2]}{2} \right\rfloor \qquad i_0 \le 2n+1 < i_N \end{aligned} \tag{3.5.3}$$

Para una señal transformada de 8 muestras, agregamos 1 elemento a la izquierda de $y[n]$ y dos a la derecha. Estos elementos se obtienen de la reflexión de la señal a la izquierda y a la derecha tomando como centro el primer y último coeficiente respectivamente.

Una vez reflejada $y[n]$, se aplican los 6 pasos de la antitransformación como se muestra en el esquema de la figura 3.5.7.

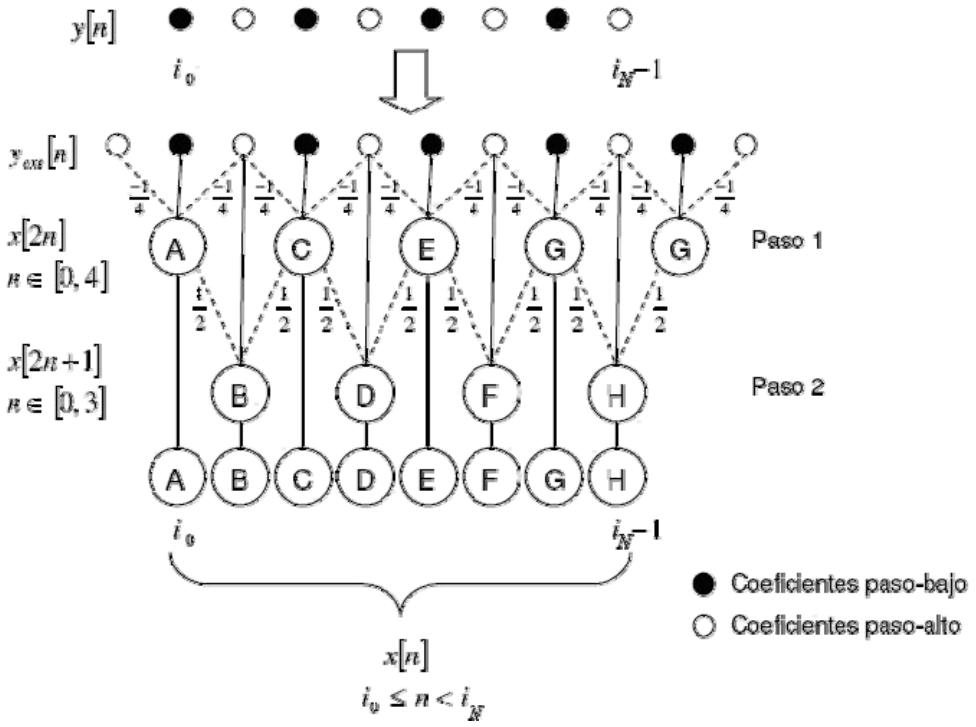


Figura 3.5.7 Diagrama de la DWT inversa con el filtro Legall 5/3 para una señal de 8 muestras

Transformada *Wavelet* Inversa usando el filtro Legall 9/7

La Transformada *Wavelet* Inversa utilizando este filtro consta de 2 pasos de escalado seguido por 4 de desplazamiento:

$$
\begin{aligned}
&paso1: x[2n] = k y_{ext}[2n] && i_0 - 3 \le 2n < i_N + 3 \\
&paso2: x[2n+1] = -(1/k)\, y_{ext}[2n+1] && i_0 - 2 \le 2n+1 < i_N + 2 \\
&paso3: x[2n] = x[2n] - (\delta[x[2n] + x[2n+2]]) && i_0 - 3 \le 2n < i_N + 3 \\
&paso4: x[2n+1] = x(2n+1] - (\gamma[x[2n] + x[2n+2]]) && i_0 - 2 \le 2n+1 < i_N + 2 \\
&paso5: x[2n] = x[2n] - (\beta[x[2n-1] + x[2n+1]]) && i_0 - 1 \le 2n < i_N + 1 \\
&paso6: x[2n+1] = x[2n+1] - (\alpha[x[2n] + x[2n+2]]) && i_0 \le 2n+1 < i_N
\end{aligned}
\qquad \textbf{(3.5.4)}
$$

La siguiente figura muestra el diagrama de flujo de la transformada *Wavelet* con el filtro Daubechies 9/7 inversa, para una señal de 8 muestras, correspondiente al ejemplo presentado para el caso de la transformada directa:

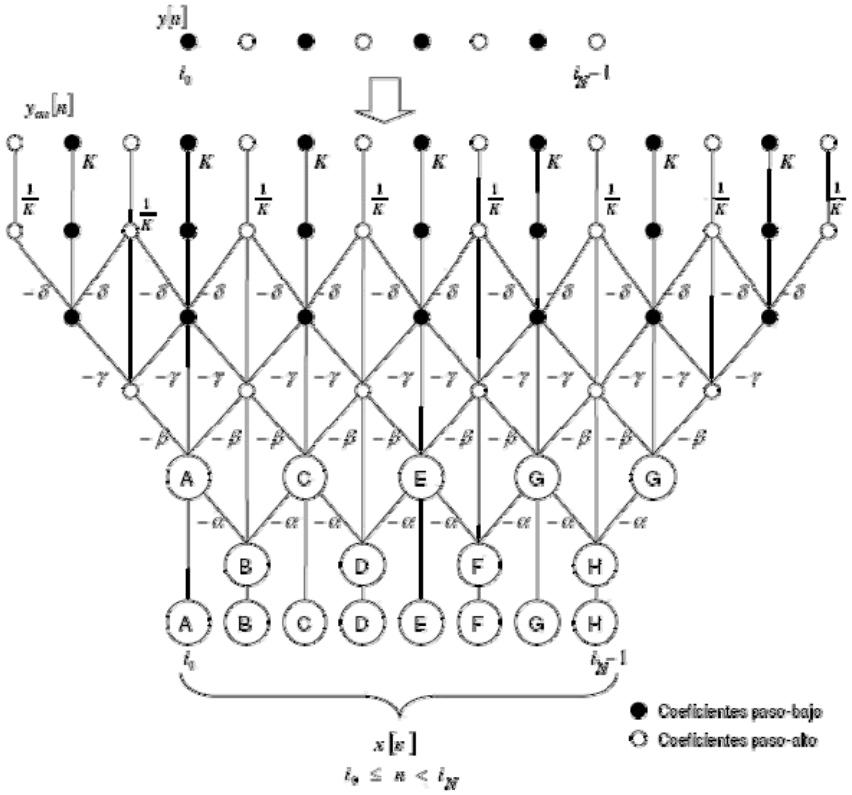


Figura 3.5.8 Diagrama de la DWT inversa con el filtro Daubechies 9/7 para una señal de 8 muestras

3.6 Aplicaciones de la transformada *wavelet*

La transformada *wavelet* es muy utilizada en un amplio campo de especialidades, tanto que sustituye muy a menudo a la tradicional transformada de Fourier, debido a que ésta brinda muy poca información para señales no estacionarios ni periódicas, como es el caso de las imágenes.

La transformada *wavelet* discreta se utiliza fundamentalmente en física, ingeniaría e informática. Sin pretender una lista exhaustiva algunas de sus aplicaciones son:

- Codificación de señales.
- Procesamiento de imágenes.
- Análisis de la sangre.
- Estudio del ADN.
- Reconocimiento de voz.
- Análisis multifractal.
- Análisis multiresolución.
- Reconocimiento de caracteres.
- Identificación y verificación de firmas.
- Análisis y diseño de sistemas de comunicaciones.
- Filtrado digital.
- Análisis espectral.
- Eliminación de ruido de señales digitales.

Por ejemplo para ésta última aplicación resulta más sencillo trabajar en el dominio de las *wavelets* que en el dominio original.

Con *wavelets*, el ruido puede ser eliminado de un gran número de tipos de señales, incluyendo aquellas con saltos, picos y otros cambios no demasiado suaves. La eliminación de ruido por *wavelets* es superior a las técnicas tradicionales, las cuales eliminan el ruido por medio de filtrados paso bajas, emborronando las zonas abruptas de la señal.

Grosso modo, el procedimiento trabaja tomando la transformada de *wavelet* de la señal con ruido, poniendo a cero los coeficientes por debajo de un cierto nivel, e invirtiendo la transformada para reconstruir la señal original con menos el ruido. El proceso de filtrar los coeficientes es bastante parecido a mantener sólo los coeficientes con mayor energía en los algoritmos de compresión de datos.

CAPÍTULO IV Algoritmo Embedded Zerotree Wavelet y estándar JPEG2000

Al inicio del presente capítulo se da una explicación de la estructura de árbol de ceros, ya que es la base para la codificación Embedded Zertotree Wavelet (EZW). Esta estructura es muy útil en señales que ha sido transformadas con wavelets, ya que aprovecha la separación de la señal en subbandas, además de que permite una transmisión progresiva.

A continuación se describe el algoritmo de compresión y codificación que consiste principalmente en dos pasos, el dominante y el subordinado, los cuales se repiten de forma iterativa. En el primer paso se eligen los coeficientes con mayor energía y en el segundo se hace un refinamiento.

Posteriormente se expone el funcionamiento del algoritmo de decodificación, que básicamente es el mismo proceso de codificación, pero de manera inversa. Tanto el algoritmo de codificación como el de decodificación incluye la transmisión y recepción progresiva de la imagen comprimida. Para ilustrar el proceso, se desarrolla un ejemplo del algoritmo aplicado a una matriz de 8 x 8

Por último se describe el estándar JPEG 2000, el cual se tomó como referencia para evaluar los resultados de esta tesis.

4.1 La estructura del árbol de ceros asociado a los coeficientes de la transformada *wavelet*

Un codificador basado en el algoritmo llamado "*embedded zerotree wavelet*" EZW [12] esta especialmente diseñado para codificar una señal a la cual se le aplicó la transformada *wavelet*. Éste fue diseñado originalmente para ser usado en imágenes (señales de dos dimensiones) transformadas, pero también puede ser usado en señales de una dimensión.

El codificador basado en EZW permite una codificación progresiva, para así comprimir una imagen en un flujo de bits con exactitud creciente, lo que significa que mientras más bits sean añadidos al flujo de salida, la imagen contendrá mejor calidad.

Además el EZW permite comprimir la información con cualquier tasa de compresión que se desee, adaptándose al ancho de banda disponible, lo que resulta en una compresión con pérdidas, pero también es posible obtener una reconstrucción perfecta con una compresión sin pérdidas.

El codificador EZW esta basado en dos importantes propiedades:

1.- Las imágenes naturales tienen generalmente un espectro paso bajas. Cuando una imagen es transformada usando la transformada *wavelet*, la energía de las subbandas decrece conforme la escala pasa de gruesa a fina (escala gruesa significa baja frecuencia), por lo tanto los coeficientes *wavelets* serán en promedio más pequeños en las subbandas de alta frecuencia que en las subbandas de baja frecuencia.

   Esto muestra que la codificación progresiva es una elección natural para compresión de imágenes con transformada *wavelet*, ya que las subbandas altas solo añaden detalle (energía de los bordes).

2.- Los coeficientes *wavelet* grandes son más importantes que los coeficientes *wavelet* pequeños, ya que los grandes contienen más energía.

Estas dos propiedades son explotadas al codificar los coeficientes *wavelet* en orden decreciente, esto se hace en múltiples iteraciones .Para cada iteración, se elige un umbral con el cual todos los coeficientes *wavelet* son comparados.

Si algún coeficiente *wavelet* es significativo, es decir, que es más grande que el umbral, éste es codificado y retirado de la imagen, pero si el coeficiente es no significativo, es decir, más pequeño que el umbral, se deja en la imagen para la siguiente iteración.

Cuando todos los coeficientes *wavelets* han sido comparados con el umbral, éste se divide a la mitad y se vuelve a repetir el mismo procedimiento, para así añadir más detalle a la imagen ya codificada. El proceso se repite hasta que todos los coeficientes *wavelet* hayan sido codificados, o hasta que algún otro criterio haya sido satisfecho (como por ejemplo, la tasa de compresión).

La dependencia de los coeficientes *wavelets* a través de las diferentes escalas es usada para codificar eficientemente partes grandes de la imagen que se encuentren por debajo del umbral actual, es en esta parte donde entra el concepto de “*zerotree*”, o árbol de ceros.

Como se mencionó anteriormente, la transformada *wavelet* transforma una señal del dominio del tiempo al dominio del *tiempo (o espacio) y escala*, lo que significa que los coeficientes *wavelet* son bidimensionales. Por lo tanto, si se quiere comprimir la señal, no sólo se debe de codificar el valor de los coeficientes, sino también su posición en el tiempo.

Cuando la señal es una imagen, entonces su posición en el tiempo equivale a su posición en el espacio.

Después de transformar una imagen usando la transformada *wavelet*, ésta se puede representar por medio de “árboles” (*trees*), debido al submuestreo que se llevó a cabo en la transformada.

Un coeficiente en una subbanda de baja frecuencia tiene cuatro descendientes en la siguiente subbanda de alta frecuencia, estos cuatro descendientes a su vez tienen cada uno cuatro descendientes en la siguiente subbanda de alta frecuencia y así sucesivamente. Entonces se obtiene un tetra-árbol (*quad-tree*), donde cada raíz posee cuatro hojas, como se ilustra en la figura 4.1.1.

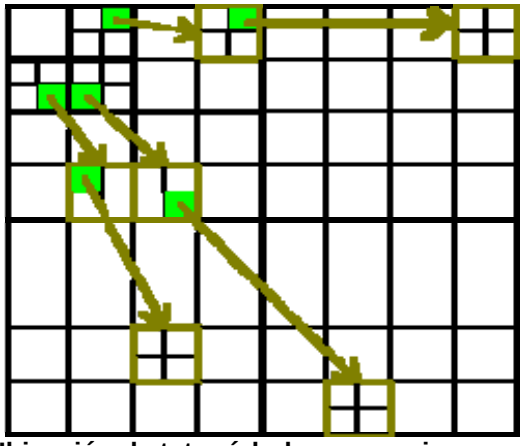


Figura 4.1.1 Ubicación de tetra-árboles en una imagen transformada

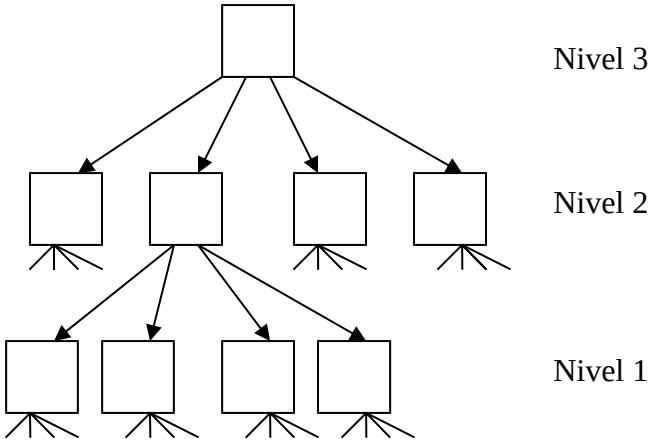


Figura 4.1.2 Representación de los tetra-árboles de acuerdo a los niveles de descomposición

Con la explicación anterior es posible dar una definición de "*zerotree*". Un "*zerotree*" es un tetra-árbol (*quad-tree*) en donde todas las hojas o nodos tienen valores iguales o menores al de la raíz. El árbol de ceros es codificado con un solo símbolo y el decodificador lo reconstruye como un tetra-árbol rellenado con ceros.

Es importante añadir a esta definición que la raíz tiene que ser un coeficiente no significativo, es decir, menor que el umbral con el cual todos los coeficientes *wavelet* estén siendo comparados en la iteración actual.

El codificador basado en el algoritmo EZW explota el "*zerotree*", basándose en la observación de que los coeficientes *wavelet* decrecen al hacerse la escala más fina. Además se asume que hay una alta probabilidad de que todos los coeficientes en un tetra-árbol sean más pequeños que un cierto umbral si su raíz es más chica que éste umbral.

Si el transmisor envía el valor del umbral y sólo manda símbolos al decodificador cuando los valores son mayores en valor absoluto al umbral actual, la señal original puede ser reconstruida casi en su totalidad.

Para obtener una reconstrucción perfecta es necesario repetir el proceso de manera sucesiva después de disminuir el valor del umbral, hasta que su valor sea menor que el valor del coeficiente más pequeño que se quiera transmitir.

Cuando son utilizados valores predeterminados de umbrales, ya no es necesario mandar su valor y de esta manera se ahorra ancho de banda.

Si la secuencia predeterminada es una secuencia de potencias de dos, se le conoce como codificación *biplanar*, ya que el umbral correspondería a los bits en un sistema binario.

La transmisión de las posiciones de los coeficientes también es muy importante, ya que sin esta información, el decodificador no será capaz de reconstruir la señal (aunque si podría reconstruir perfectamente el flujo de bits transmitidos).

Como ya se menciono, el codificador EZW, realiza un barrido predefinido para codificar la posición de los coeficientes *wavelet*. Con el uso del “zerotree”, muchas posiciones son codificadas implícitamente.

Es posible realizar muchos barridos de diferente orden, siempre y cuando las subbandas de baja frecuencia sean completamente barridas antes de que se pase a las subbandas superiores, como el barrido *Raster* y el barrido *Morton.*

El barrido que utilizamos en este trabajo es el barrido Morton y se muestra en la figura 4.1.3.

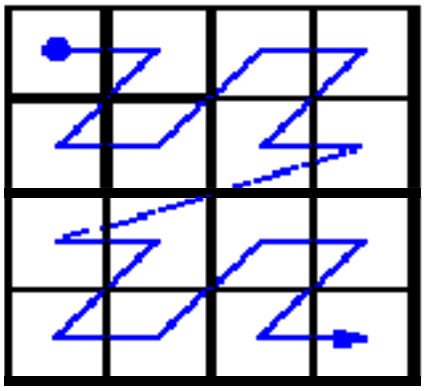


Figura 4.1.3 Barrido Morton.

4.2 Funcionamiento del algoritmo de compresión y codificación con transmisión progresiva

El flujo de salida del algoritmo EZW deberá de comenzar con algún tipo de información para sincronizar al decodificador, ésta puede ser el tipo de la *wavelet* que se utilizó para el análisis, además del valor del umbral inicial así como el tamaño de la imagen original. El primer paso en el algoritmo del codificador EZW, es determinar el umbral inicial, que en el caso del codificación biplanar, el umbral inicial esta dado por t_0:

$$t_0 = 2^{[\log_2(MAX|\gamma(x,y)|)]} \qquad (4.1)$$

Donde $MAX|\gamma(x,y)|$ es el valor del máximo coeficiente *wavelet* en la imagen. Ya teniendo el umbral inicial se entra en un ciclo principal, el cual se muestra en el diagrama de flujo de la figura 4.2.1.

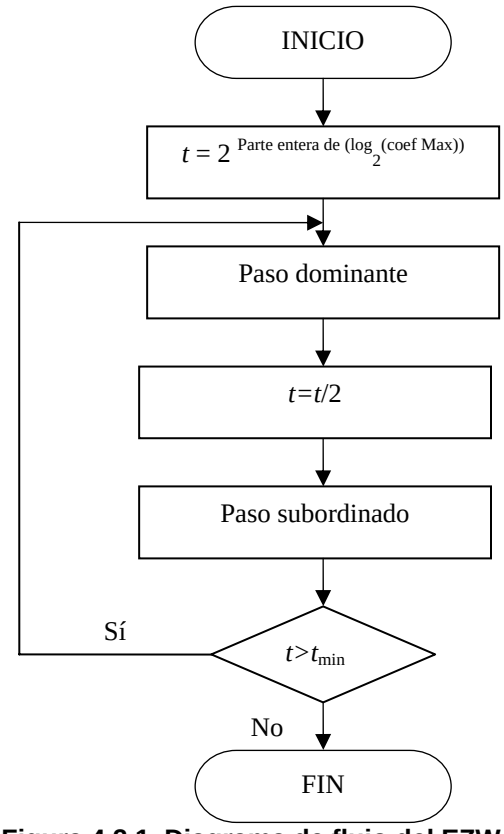


Figura 4.2.1. Diagrama de flujo del EZW

Como se puede observar, existen dos pasos principales por los cuales debe de pasar la matriz de coeficientes wavelets.

Paso dominante

En el “paso_dominante”, la imagen transformada es barrida y por cada coeficiente habrá un símbolo de salida, de acuerdo a la siguiente convención:

- **P** (positivo): cuando el coeficiente es mayor en valor absoluto que el umbral *t*, y de signo positivo.
- **N** (negativo): cuando el coeficiente es de signo negativo, y mayor en valor absoluto que el umbral *t*.
- **T** (raíz de *zerotree*): cuando el coeficiente y todos sus descendientes son más pequeños que el umbral. Por lo tanto *sus descendientes ya no serán barridos*.
- **Z** (cero aislado): cuando el coeficiente es no significativo con respecto al umbral, pero al menos uno de sus coeficientes descendientes sí es significativo.

Para determinar si un coeficiente es la raíz de un “zerotree”, o un cero aislado, se deberá de barrer todo el tetra-árbol, es decir, la raíz y sus descendientes. Para prevenir que se codifiquen coeficientes que ya han sido identificados como “zerotrees”, se deberá de tener un seguimiento de éstos marcándolos.

Finalmente todos los coeficientes que se encontraron significativos, son extraídos de la matriz de coeficientes y guardados con todo y su signo en la lista de coeficientes significativos. En su lugar se colocarán ceros, de esta manera se evita que sean codificados nuevamente.

La figura 4.2.2 muestra el diagrama de flujo del paso dominante.

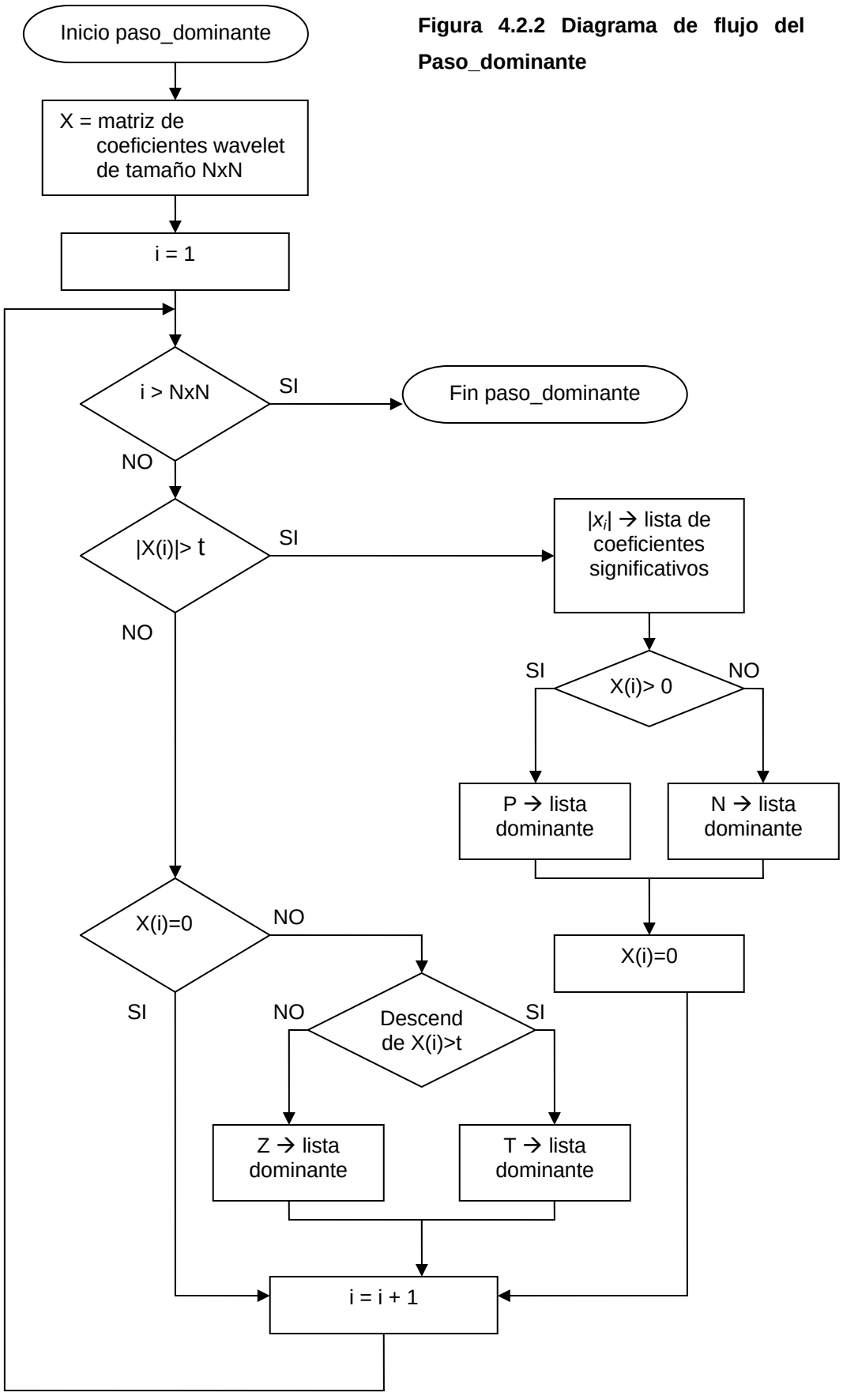


Figura 4.2.2 Diagrama de flujo del Paso_dominante

Paso subordinado

La segunda función es el “paso_subordinado”, en donde se lleva a cabo el refinamiento, el cual consiste en la cuantización de los coeficientes significativos, con lo cual se realiza una compresión.

El primer paso es especificar el intervalo de [t, 2t), note que el tamaño o rango del intervalo es de t. Después se define el umbral que lo divida a la mitad t+t/2, de modo que el coeficiente significativo pueda encontrarse ya sea en la mitad inferior [t, t+t/2), o en la superior [t+t/2, 2t). Se omite el signo de los coeficientes y en la lista subordinada se guarda un ‘1’ si se encuentra en la mitad superior del intervalo, y un ‘0’ si está en la mitad inferior.

El decodificador deberá tener el símbolo de la lista subordinada y el umbral de los intervalos, para que sea capaz de determinar el intervalo en el cual está situado dicho coeficiente y con esto reconstruir su valor.

Esta lista esta ordenada de tal manera que los coeficientes mayores sean los primeros que se transmitan. Cada vez que termina el paso dominante y subordinado, se envían la lista dominante y la subordinada con los coeficientes de las iteraciones pasadas al principio.

La codificación con el algoritmo EZW termina cuando el umbral llega a su valor mínimo, para valores enteros, este valor del umbral es cero, o bien cuando la tasa de transmisión deseada haya sido alcanzada.

A continuación se muestra el diagrama de flujo del paso_subordinado:

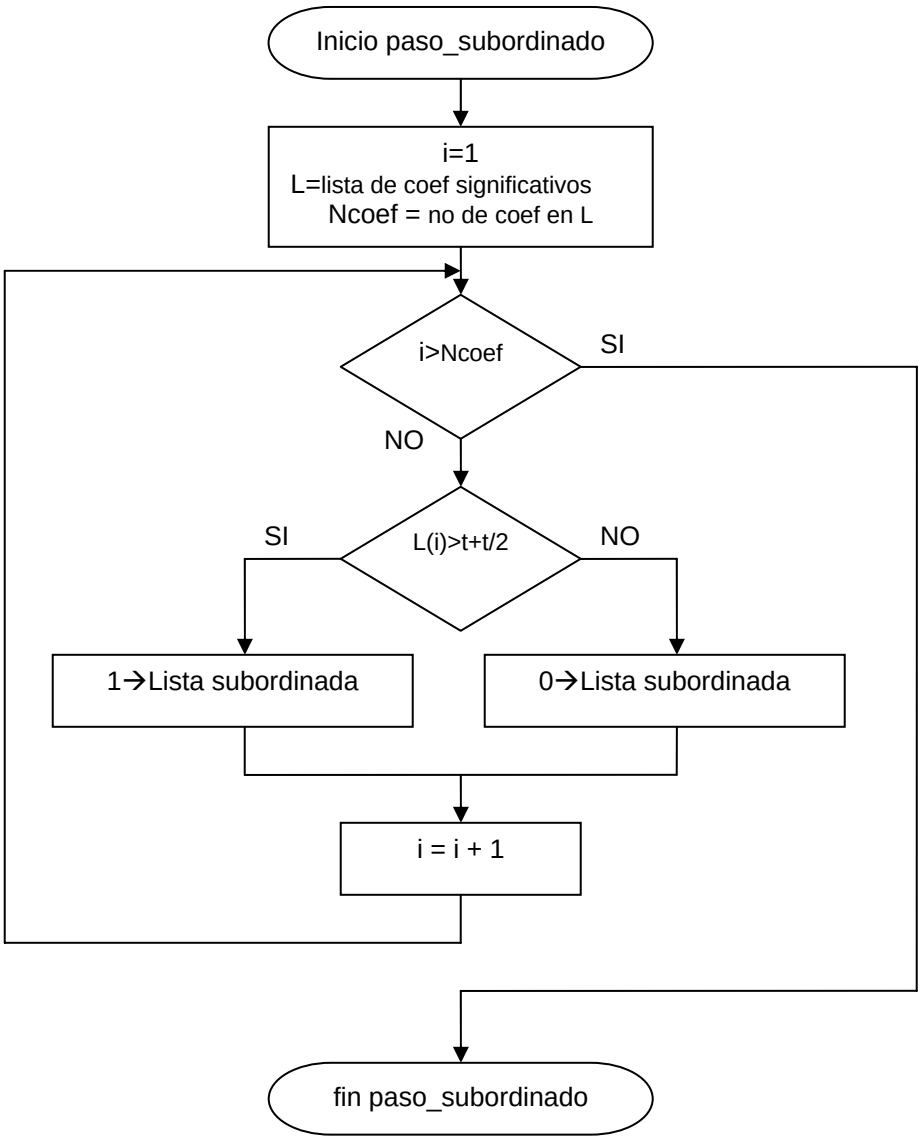
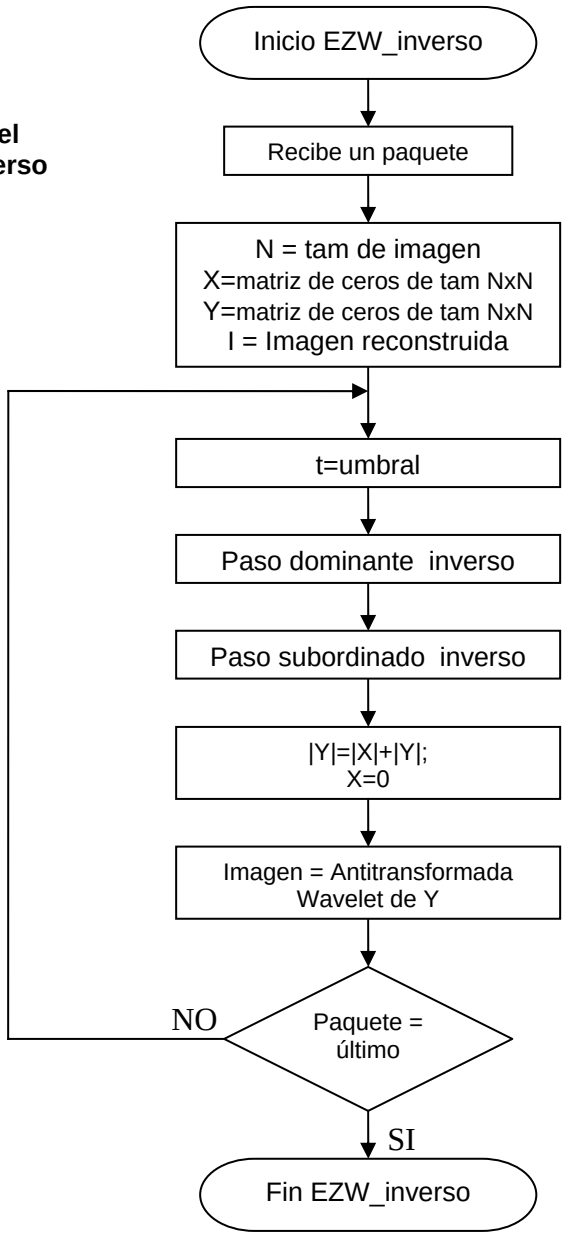


Figura 4.2.3. Diagrama de flujo del Paso_subordinado

4.3 Funcionamiento del algoritmo de decodificación

El decodificador sigue el proceso inverso al codificador. Éste debe conocer, por lo menos, el tamaño de la imagen, la lista dominante, la subordinada y el umbral utilizado. Cuando el primer paquete es recibido, establece el tamaño de la imagen que va a reconstruir y obtiene el umbral inicial t. Después realiza el paso subordinado_inverso y luego el dominante_inverso como se muestra en el siguiente diagrama.

Figura 4.3.1.
Diagrama de flujo del algoritmo EZW_inverso



En un proceso del algoritmo EZW inverso, no es necesario antitransformar después de cada iteración, ya que se puede esperar a que estén listos todos los coeficientes decodificados y al final antitransformar para obtener la imagen. Pero en el caso de la transmisión progresiva, se debe efectuar la transformación inversa (antitransformación) una vez que el paquete recibido es decodificado.

Para este paso se necesita una matriz auxiliar Y, en la cual se irán guardando los coeficientes decodificados después de cada iteración, y una lista L en donde se irá guardando la posición de los coeficientes que se han encontrado significativos.

Paso dominante inverso

El decodificador utiliza el dato del umbral como referencia en el paso dominante inverso. En seguida obtiene un elemento de la lista dominante:

- Si es P ó N, el decodificador interpreta que el coeficiente es significativo, por tanto le asigna el valor del umbral con signo positivo o negativo según sea el caso y guarda en la lista L la posición del coeficiente.

- Si el elemento de la lista es Z, quiere decir que el coeficiente es no significativo, por lo que le da el valor de 0.

- Si el elemento es T, el decodificador lo interpreta como un árbol de ceros, por lo tanto al coeficiente y a sus descendientes les da el valor de 0.

El orden en que el decodificador va llenando la matriz de coeficientes es el mismo que el del barrido tipo “Morton”. También debe saber si el lugar que va a ocupar no es parte de un árbol de ceros. Para eso usamos una matriz auxiliar A del mismo tamaño que se inicializará en ceros, y se colocará un 1 cuando el coeficiente ya fue barrido.

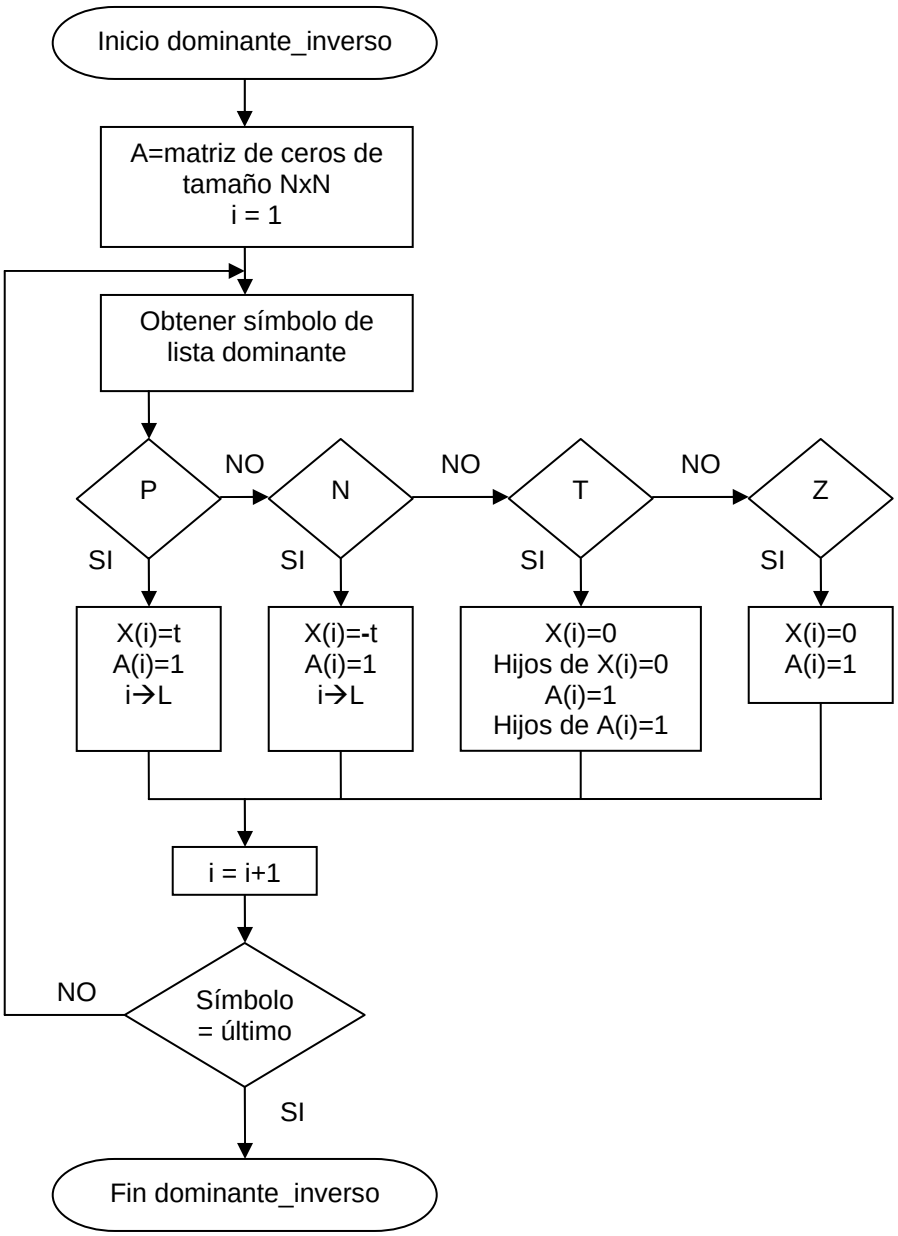


Figura 4.3.2 Diagrama de flujo del paso dominante_inverso

Ahora tenemos una matriz de coeficientes que se refinará con el paso subordinado_inverso el cual se explica a continuación.

Paso subordinado_inverso

El decodificador identifica dos intervalos: el inferior [t, t+t/2) y el superior [t+t/2,2t). Obtiene de la lista L la posición del coeficiente que se va a refinar y toma un símbolo de la lista subordinada:

- Si es uno, quiere decir que el coeficiente significativo está en el intervalo superior, por lo que le suma t/2 en magnitud, por lo que conserva su signo.

- Si el símbolo de la lista subordinada es 0, entonces el coeficiente se mantiene sin cambio.

Ahora obtiene de la lista L la posición del siguiente coeficiente y repite estos pasos hasta que ya no haya elementos en la lista L, ni en la subordinada.

En la figura 4.3.3 se observa el diagrama de flujo de este proceso.

La matriz resultante X será sumada **sólo en magnitud** a la matriz Y, es decir que a los coeficientes negativos de Y se les resta los de X y a los positivos se les suma. La matriz Y será antitransformada usando la misma *wavelet* y el mismo número de niveles que usó el codificador.

Cuando llega el siguiente paquete, se repite el paso dominante inverso y subordinado inverso hasta que llegue el último paquete.

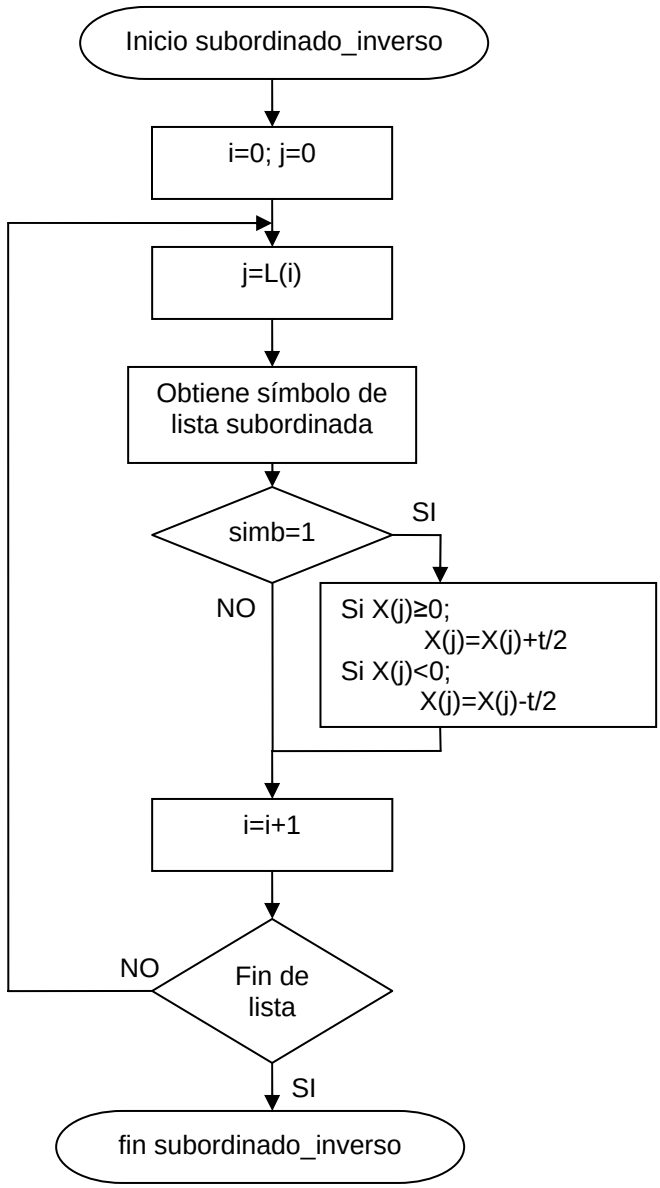


Figura 4.3.3 Diagrama de flujo del paso subordinado inverso

4.4 Ejemplo de codificación y decodificación con EZW

Para reforzar la explicación del proceso de codificación y decodificación, se verá un ejemplo de la manera en que el algoritmo EZW es llevado a cabo.

Codificación

Para efectuar este ejemplo, se propone una matriz de coeficientes, supuestamente resultado de transformar una imagen.

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Figura 4.4.1 Matriz de coeficientes resultado de una imagen transformada

Dado a que el coeficiente mayor es 63, el umbral inicial se calcula de la siguiente manera:

$$t_0 = 2^{\text{Parte entera de } (\log_2(63))} = 2^5 = 32$$

Comenzamos con el paso dominante barriendo los coeficientes según el barrido tipo “Morton”:

|63|>32, es significativo (mayor que el umbral) y positivo, por tanto, se coloca una P en la lista dominante;

|-34|>32, como es significativo y de signo negativo, se coloca una N en la lista dominante;

|-31|<32, es no significativo, pero entre sus descendientes hay un coeficiente significativo (el 47), por lo tanto, se coloca una Z en la lista dominante

|23|<32, es no significativo, y sus descendientes tampoco son significativos, por lo tanto, se coloca una T en la lista dominante y **sus descendientes ya no serán barridos en la iteración actual.**

Así sucesivamente: 49→P, 10→T, 14→T, -13→T, 15→T, 14→Z, -9→T, -7→T, 7→T, 13→T, 3→T, 4→T, -1→T, 47→P, -3→T, 2→T.

De manera que la primera lista dominante D1 queda:

D1: PNZTPTTTTZTTTTTTTPTT

Todos los coeficientes que se encontraron significativos se sustituyen por un 0 en la matriz de coeficientes para que no sean barridos nuevamente.

Ahora comienza el paso subordinado. Del intervalo [32,64) se obtienen dos subintervalos: [32,48) y [48,64). De la lista dominante sabemos que hay solo cuatro coeficientes significativos, los cuales, omitiendo el signo, el 63 está en el intervalo superior (se coloca un 1 en la lista subordinada), el 34 en el inferior (se coloca un 0 en la lista subordinada), etcétera. De manera que la lista subordinada queda:

S1: 1010

Esto concluye una iteración del algoritmo, por lo que puede ser enviado el primer paquete con el tamaño de la imagen, el umbral inicial y las dos listas.

Para la siguiente iteración, el umbral se divide a la mitad: t_1=16 y obtenemos la lista dominante con este umbral.

0	0	0	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	0	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Figura 4.4.2 Matriz de coeficientes en la segunda iteración en la codificación

D2: ZTNPTTTTTTTT

Para el paso subordinado tenemos los siguientes intervalos de tamaño 16: [16,32), [32,48), [48,64), con sus umbrales correspondientes: 24, 40 y 56. Se toman los coeficientes significativos de la barrida anterior (63, -34, 49, 47), y omitiendo el signo se tiene:

63>56 →1

34<40→0

49<56→0

47>40→1

Se consideran después los coeficientes significativos de la nueva lista dominante (-31, 23):

31>24→1

23<24→0

La segunda lista subordinada resulta de la siguiente manera:

S2:100110

Al término de la iteración se puede enviar un segundo paquete y así sucesivamente.

Las iteraciones completas quedarían:

Umbral = 32
D1: PNZTPTTTTZTTTTTTTPTT
S1: 1010
Umbral = 16
D2: ZTNPTTTTTTTT
S2: 100110
Umbral = 8
D3:
ZZZZZPPNPPNTTNNPTPTTNTTTTTTTTPTTTPTTTTTTTTTPTTTTTTTTTTTT
S3: 10011101111011011000
Umbral = 4
D4:
ZZZZZZZTZTZNZZZZPTTPTPPTPNPTNTTTTTPTPNPPPPTTTTTPTPTTTPNP
S4: 1101111101100100000111011010000100101011100
Umbral =2
D5:
ZZZZZTZZZZZTPZZZTTPTTTTNPTPPTTPTTTNPPNTTTTPNNPTTPTTPPTTT
S5: 10111100110100010111110101101100100000000110110110011000111
Umbral=1
D6: ZZZTTZTTTZTTTTTNNTTT
El último paso puede prescindir de la lista subordinada ya que sólo queda el 1 y el 0.

Decodificación

Del primer paquete se obtiene el tamaño de la imagen y el umbral t_0=32.

Con el paso dominante inverso decodificamos la lista dominante PNZTPTTTTZTTTTTTTPTT y se colocan los coeficientes en la matriz X de acuerdo al orden del barrido tipo “Morton”:

El primer elemento de la lista es P, por lo tanto se coloca en la matriz un 32 y se guarda en la lista L su posición (0,0). El siguiente elemento es N, y se coloca un -32 en la matriz y la posición (0,1) en la lista L. El siguiente es Z, por lo que se coloca un 0 en la matriz y nada en la lista L. El siguiente elemento de la lista es T, por lo tanto se coloca un árbol de ceros en la matriz, es decir que de acuerdo al barrido “Morton”, se coloca un 0 en la posición (0,0) y también se colocan ceros en el lugar de los descendientes del coeficiente. En la matriz auxiliar A, se colocan 1 en el lugar de los coeficienes que ya hayan sido barridos. Y así sucesivamente. Al término de este paso dominante inverso tenemos la siguiente matriz:

32	**-32**	**32**	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**32**	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 4.4.3 Matriz X después del paso dominante inverso (1ª iteración)

Y en la lista L tenemos (0,0), (0,1), (0,2), (4,3)

Ahora se realiza el paso subordinado inverso con la lista subordinada 1010.

Sabemos que el umbral es 32, la mitad sería 16. Obtenemos los elementos de la lista L y de la lista subordinada

L(1)=(0,0) Lista_subordinada(1) = 1

Por lo tanto, al coeficiente en (0,0) le sumamos 16

L(2)= (0,1) Lista_subordinada(1) = 0

Al coeficiente en (0,1) no se le suma nada.

Pasando por todos los elementos de las listas, el resultado es:

48	**-32**	**48**	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**32**	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 4.4.4 Matriz X después del paso subordinado inverso (1ª iteración)

Esta matriz X es sumada (en magnitud) a la matriz auxiliar Y, que en un inicio es una matriz de ceros. La reconstrucción de la matriz Y es una aproximación que se irá aproximando más a los coeficientes de la imagen original conforme lleguen más paquetes.

Cuando llegue el siguiente paquete se realiza el paso dominante con el umbral 16 y la lista dominante D2: ZTNPTTTTTTTT

0	0	0	0	0	0	0	0
-16	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 4.4.5 Matriz X después del paso dominante inverso (2ª iteración)

Se agrega la posición de los nuevos coeficientes significativos a la lista L:

(0,0), (0,1), (0,2), (4,3), **(1,0), (1,1)**

Ahora realizamos el paso subordinado con la lista S2: 100110. Si el elemento es 1 le sumamos 16/2=8

El resultado es:

8	**0**	**0**	0	0	0	0	0
-**24**	**16**	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**8**	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 4.4.6 Matriz X después del paso subordinado inverso (2ª iteración)

Nuevamente esta matriz es sumada (en magnitud) a la matriz Y para que sea antitransformada. Es importante conservar el signo de los coeficientes. El resultado es el siguiente:

56	-32	48	0	0	0	0	0
-24	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	40	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 4.4.7 Matriz Y en la 2ª iteración.

El proceso termina con el último paquete recibido. Entonces es cuando se obtiene la reconstrucción final de la imagen.

4.5 El estándar JPEG 2000

El estándar de compresión de imágenes digitales JPEG2000 es un estándar que fue desarrollado por el grupo JPEG (Joint Picture Expert Group).

Algunas de sus características más importantes son:

- Se encuentra basado en las *wavelets*.
- Se puede obtener compresión con, o sin pérdidas, utilizando transformadas reversibles de enteros a enteros, y de reales a reales, respectivamente.
- Posibilidad de transmitir progresivamente, ya sea por resolución, regiones de interés, de acceso aleatorio a una zona específica de la imagen, sin que se tenga que decodificar la imagen completa.

- Realiza la codificación de datos transformados mediante el uso de técnicas de codificación por planos de bits.
- El codificador de entropía usa un codificador aritmético binario adaptable (MQ).
- División de la imagen en partes más pequeñas para que su codificación sea más rápida.

El estándar JPEG2000 esta conformado por distintas partes, en la primera parte se definen el codificador y el decodificador básico, en la segunda y tercera parte se describen extensiones del codificador básico que son de gran utilidad para aplicaciones específicas. [14]

La estructura de principio del codificador y del decodificador se muestra a continuación:

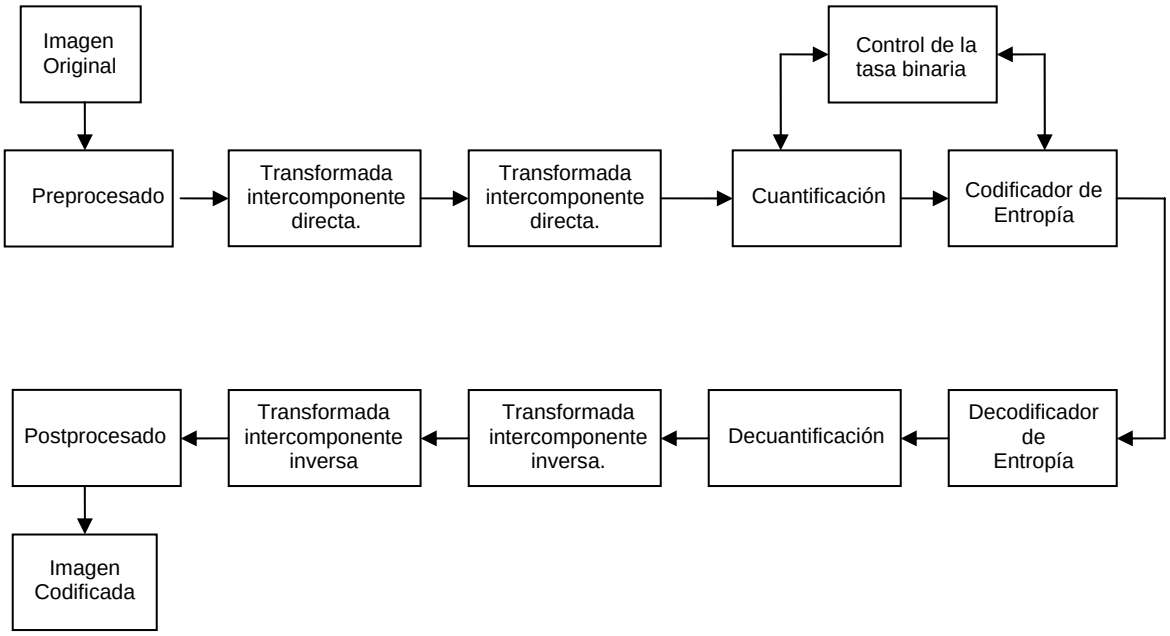


Figura 4.5 Diagrama de bloques del proceso de codificación y decodificación JPEG2000

Como se puede observar en la figura anterior la estructura del decodificador es casi la inversa al codificador, exceptuando la parte del control de la tasa binaria, ya que en esta parte los bloques de la imagen son codificados de manera independiente.

Teniendo una imagen que se quiera comprimir y transmitir el primer paso que se debe de realizar es el **Preprocesado,** en donde se debe se asegurar que la señal de entrada se encuentre centrada en cero, de no ser así se deberá de realizar un desplazamiento del nivel de “continua” igual al la mitad del valor máximo que puedan tomar las muestras de entrada.

Las muestras de entrada pueden ser con signo, o no, si la señal tiene signo entonces ya se tendrá un rango dinámico centrado en cero.

Actualmente JPEG 2000 no está ampliamente admitido por los programas de visualización de páginas web. Muchos diseñadores no tienen intención de incluirlo debido a su escaso uso y gran número de patentes que tiene.

Independientemente de ello, el estándar JPEG2000 es utilizado como referencia en este trabajo. En el siguiente capítulo se muestran los resultados del programa implementado en el presente trabajo comparado con resultados del estándar JPEG2000.

CAPÍTULO V Resultados y conclusiones

En este capítulo se plasman los resultados de simulaciones y evaluaciones del codificador diseñado, mismos que se comparan con los resultados obtenidos de la norma JPEG2000.

Gracias a estos resultados se obtienen una serie de conclusiones con respecto al cumplimiento del objetivo. A su vez, estas conclusiones dan lugar a diferentes trabajos que quedan pendientes para un futuro.

Después de muchas pruebas de desempeño el grupo JPEG decidió escoger dos funciones *wavelet* para el estándar JPEG2000, una de ellas para codificación sin perdidas LeGall (5/3), y la otra para codificación con pérdidas Daubechies (9/7). Dada esta selección internacional y para situarnos en el mismo contexto, en esta tesis decidimos hacer nuestro estudio utilizando las mismas funciones.

5.1 Resultados

Con el trabajo que se desarrolló en esta tesis podemos comparar diferentes resultados entre las dos *wavelets* empleadas. Recordemos que la *wavelet* LeGall (5/3) es reversible, mientras que la Daubechies (9/7) no lo es. Por lo tanto es de suponer que con Daubechies obtendremos una mayor compresión, pero con LeGall obtendremos una mejor calidad.

En las siguientes tablas se muestran los resultados del análisis y síntesis con diferentes parámetros, para la imagen “Lena256.pgm”.

Los parámetros que se varían son:

- Tipo de *wavelet*: LeGall 5/3 y Daubechies 9/7
- Niveles de análisis: del 1 al 8
- Umbral: 256, 128, 64, …, 1

Los resultados para comparación, medidos son:

- Costo de entropía (cantidad de información que se tiene que enviar) [Adimensional].
- Medida de calidad PSNR (Razón señal pico a ruido) [dB].
- Tasa de compresión: relación entre el tamaño del archivo original y el comprimido. [adimensional].
- Tiempo de procesamiento: tiempo que tarda el programa en analizar, o sintetizar, una imagen [s].

También se aplicará el proceso de compresión a imágenes de características diferentes para generalizar un poco más los resultados.

Tabla 5.1. Costo .vs. Niveles de análisis. con un Umbral de 16

Nivel	1	2	3	4	5	6	7	8
LeGall(5/3)	2.7715	0.8127	0.3349	0.2234	0.1981	0.1921	0.1906	0.1902
Daubechies(9/7)	2.7313	0.7780	0.3016	0.1888	0.1627	0.1565	0.1550	0.1546

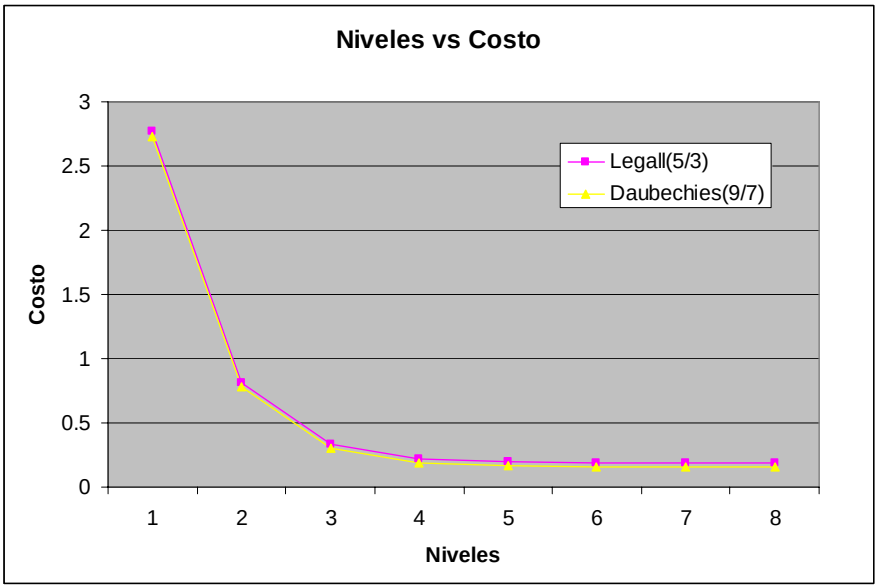


Figura 5.1. . Costo .vs. Niveles de análisis. con un Umbral de 16

Tabla 5.2. Umbral .vs. PSNR, con 5 niveles de análisis

Umbral	256	128	64	32	16	8	4	2
LeGall (5/3)	10.8886	14.9525	16.1043	19.4328	27.3187	35.9176	40.4136	41.3971
Daubechies (9/7)	14.7943	14.7943	13.8846	18.2566	25.8662	31.404	38.6537	44.9017

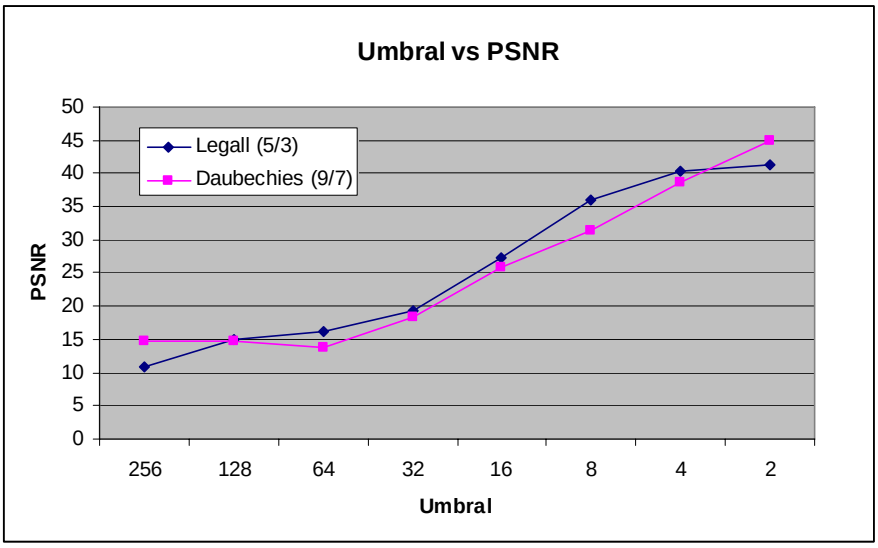


Figura 5.2. Umbral .vs. PSNR, con 5 niveles de análisis

Tabla 5.3. PSNR .vs. Niveles de análisis. con diferentes umbrales

Umbral 128

Nivel	1	2	3	4	5	6	7	8
LeGall(5/3)	11.743	16.4862	17.7119	16.2623	14.9525	13.2903	13.6628	13.6628
Daubechies(9/7)	17.9139	17.509	16.9083	15.5411	14.7943	13.4009	13.5199	13.5199

Umbral 32

Nivel	1	2	3	4	5	6	7	8
LeGall(5/3)	28.9204	23.3134	22.5984	19.7553	19.4328	19.2048	21.0395	19.7166
Daubechies(9/7)	23.2665	21.0041	20.6948	17.366	18.2566	20.0028	17.2674	18.2363

Umbral 2

Nivel	1	2	3	4	5	6	7	8
LeGall(5/3)	52.7243	50.7681	49.7831	49.417	49.2219	49.1025	49.1025	49.1025
Daubechies(9/7)	49.4148	47.5726	46.3131	45.3364	44.9017	44.0089	42.515	42.5239

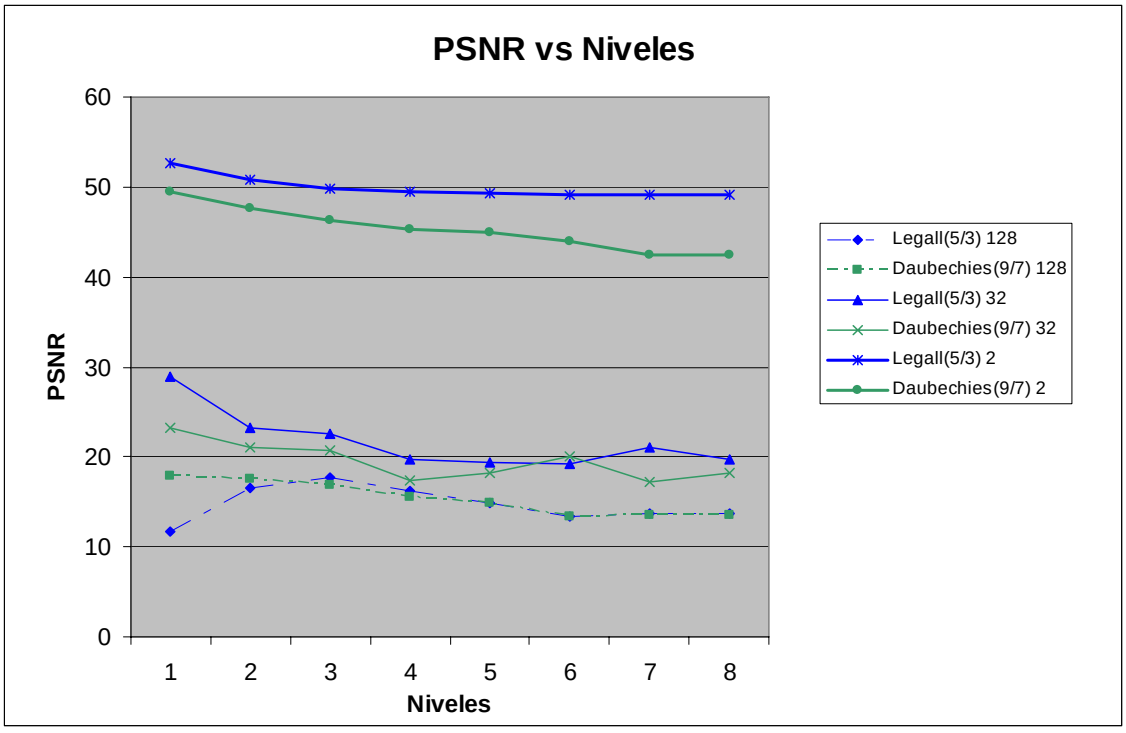


Figura 5.3. PSNR .vs. Niveles de análisis. con diferentes umbrales

Tabla 5.4. Tamaño y Tasa de compresión .vs. Calidad. para diferentes niveles de análisis

1 nivel

Umbral	128	64	32	16	8	4	2
LeGall (5/3)	13 bites	7.1 kb	16.4 kb	22.6 kb	30.1 kb	38.6 kb	53 kb
Daubechies (9/7)	13 bites	6.8 kb	16.2 kb	22.7 kb	30.4 kb	39 kb	52.2 kb
Tasa de compresión promedio	1:4923	1:9.14	1:3.92	1:2.82	1:2.11	1:1.64	1:1.21
PSNR LeGall	**11.743**	24.3661	**28.9204**	32.1554	39.5863	45.225	52.7243
PSNR daubechies	17.9139	23.4585	23.2665	**27.6651**	38.339	43.3441	49.4148

3 niveles

Umbral	128	64	32	16	8	4	2
LeGall (5/3)	20 bites	558 b	1.8 kb	4.8 kb	10 kb	18.4 kb	32.9 kb
Daubechies (9/7)	13 bites	462 b	1.6 kb	4.2 kb	9.1 kb	17.1 kb	30.2 kb
Tasa de compresión promedio	1:3878.78	1:125.49	1:37.64	1:14.22	1:6.70	1:3.60	1:2.02
PSNR LeGall	17.7119	18.9354	22.5984	**30.0644**	37.1906	42.6091	49.7831
PSNR daubechies	16.9083	17.2816	20.6948	**26.4648**	31.2804	40.2717	46.3131

5 niveles

Umbral	128	64	32	16	8	4	2
LeGall (5/3)	20 bites	152 b	1.1 kb	3.9 kb	9 kb	17.4 kb	31.9 kb
Daubechies (9/7)	13 bites	55 b	761 b	3.2 kb	8 kb	16 kb	29 kb
Tasa de compresión promedio	1:3878.78	1:618.35	1:68.78	1:18.02	1:7.52	1:3.83	1:2.10
PSNR LeGall	14.9525	16.1043	19.4328	**27.3187**	35.9176	41.3369	49.2219
PSNR daubechies	14.7943	13.8846	18.2566	**25.8662**	31.404	38.6537	44.9017

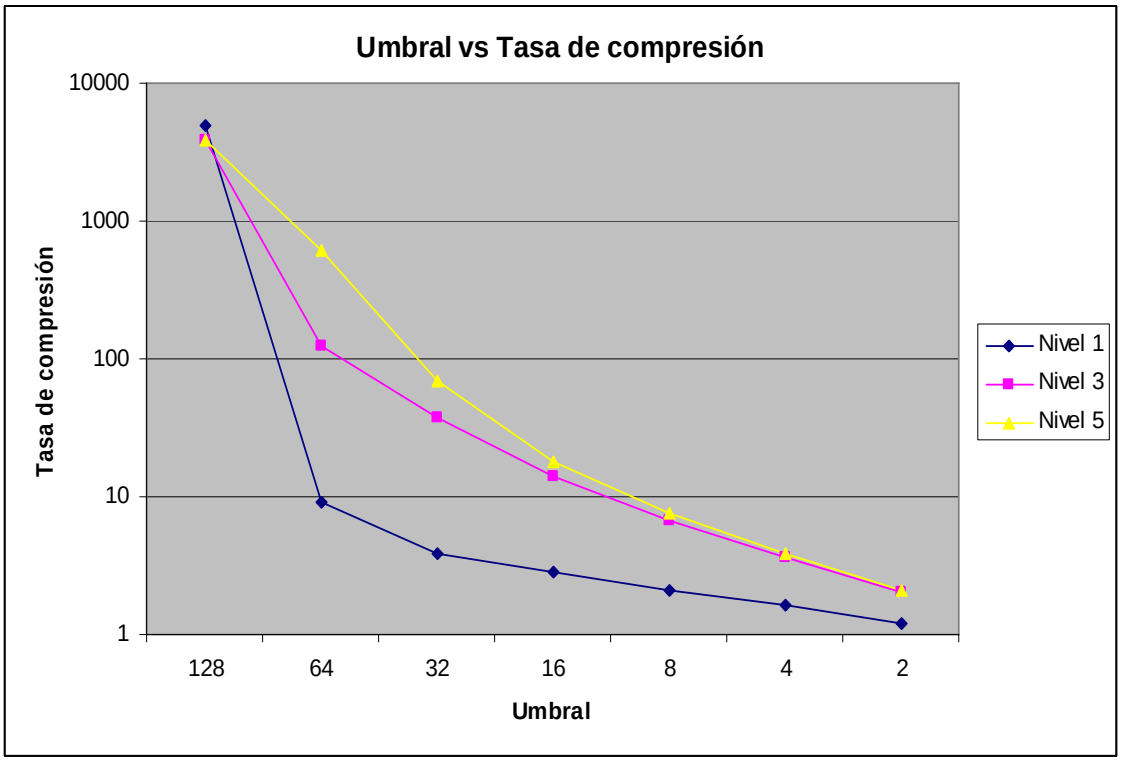


Figura 5.4.Umbral vs Tasa de Compresión.

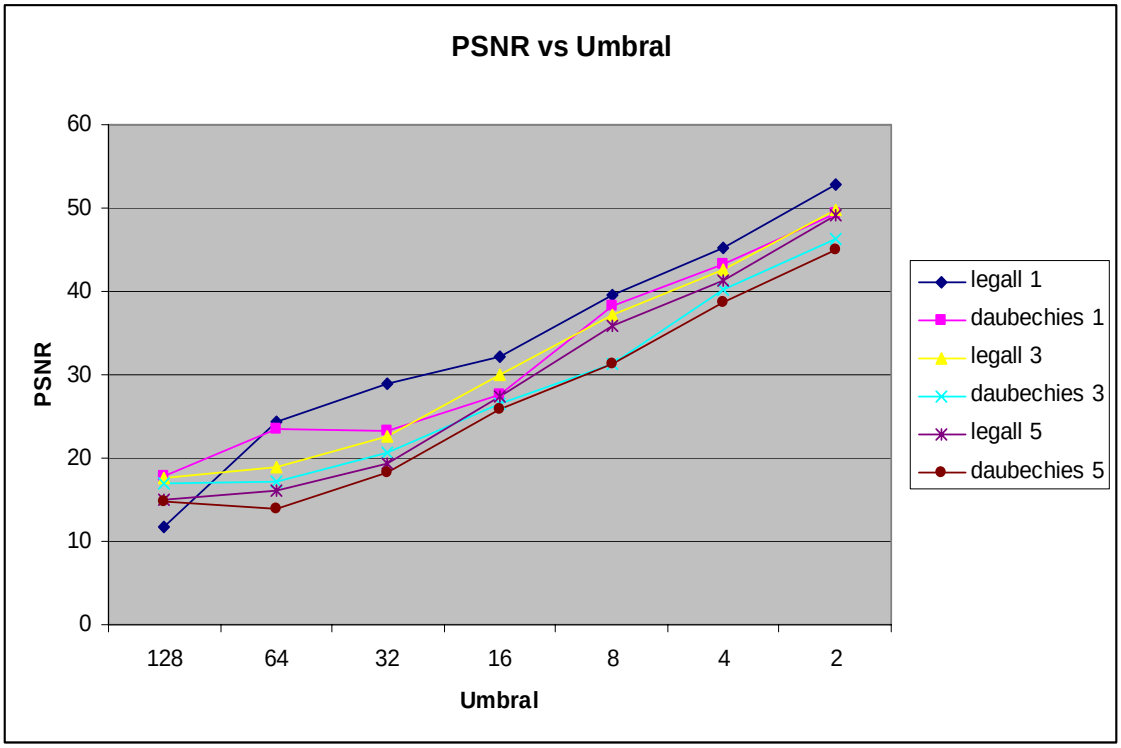


Figura 5.5. PSNR vs Umbral.

Tabla 5.5. Medición del tiempo de procesamiento con la función wavelet LeGall 5/3

Compresión:

Nivel	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Tiempo	**3’ 37’’**	**2’31’’**	**2’22’’**	**2’20’’**	**2’20’’**	**2’21’’**	**2’22’’**	**2’20**

Descompresión con 5 niveles:

Umbral	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Tiempo	**2’’**	**4’’**	**4’’**	**7’’**	**14’’**	**49’’**	**2’32’’**	**2’30’’**

Imágenes

Costo .vs. Niveles de análisis, con un Umbral de 16

LeGall 5/3

Nivel 1

Daubechies 9/7

Nivel 5

Umbral ,vs, PSNR, con 5 niveles de análisis

LeGall 5/3

Umbral 256

Daubechies 9/7

Umbral 256

LeGall 5/3

Umbral 32

Umbral 2

Daubechies 9/7

Umbral 32

Umbral 2

PSNR vs Niveles con umbral 32

LeGall 5/3

Nivel 1

Nivel 3

Daubechies 9/7

Nivel 1

Nivel 3

Nivel 5

Nivel 5

Tamaño y Tasa de compresión .vs. Calidad para diferentes niveles de análisis

LeGall 5/3

Nivel 1, umbral 32

Tasa de compresión: 1:3.92

PSNR= 28.9204 [dB]

Nivel 3, umbral 16

Tasa de compresión: 1:14.22

PSNR= 30.0644 [dB]

Nivel 1, umbral 128

Tasa de compresión: 1:4923

PSNR= 11.743 [dB]

Nivel 5, umbral 16

Tasa de compresión: 1:18.02

PSNR = 27.3187 [dB]

Daubechies 9/7

Nivel 1, umbral 16

Tasa de compresión: 1:2.82

PSNR= 27.6651 [dB]

Nivel 3, umbral 16

Tasa de compresión: 1:14.22

PSNR= 26.4648 [dB]

Nivel 5, umbral 16

Tasa de compresión: 1:18.02

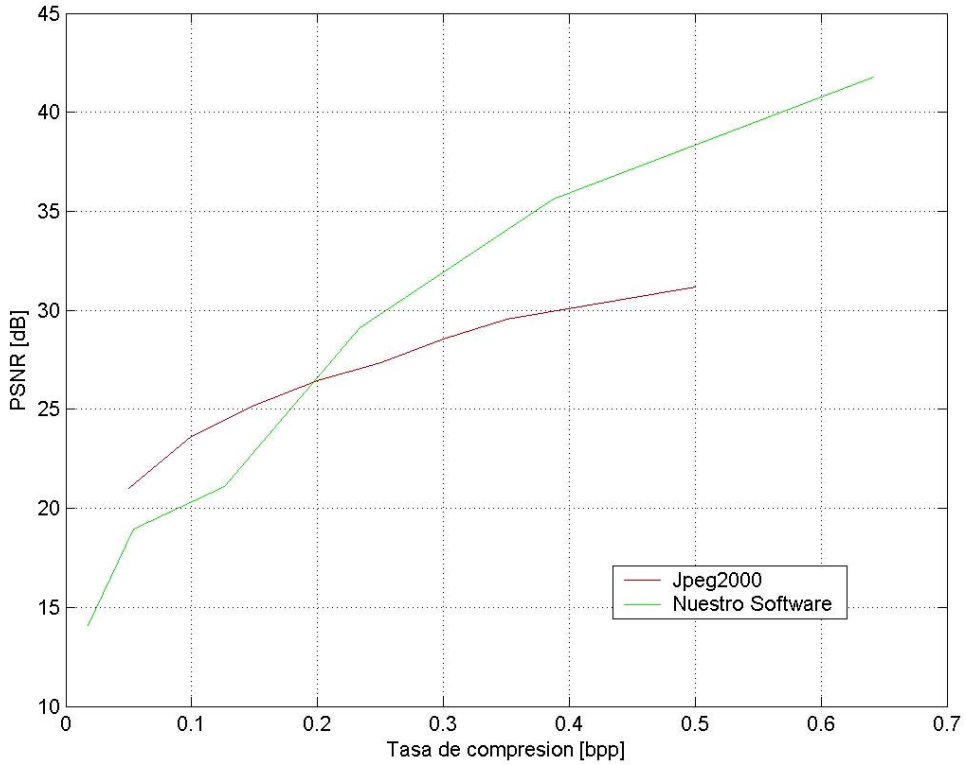
PSNR= 25.8662 [dB]

Tabla 5.6 Comparación con JPEG2000

Tomamos como referencia la tabla 1 "*PSNR .vs. bit rate for the images cameraman*" del artículo "*Image compression using an edge adapted redundant dictionary and wavelets"* [22]. Usamos la imagen cameraman.pgm

Tasa (bpp)	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.50
Jpeg2000	21.00	23.63	25.23	26.45	27.38	28.53	29.57	31.16

Tasa (bpp)	0.018	0.054	0.126	0.234	0.387	0.642
Nuestro Software	14.07	18.97	21.115	29.1135	35.60	41.78



Imágenes reconstruidas con nuestro *software*:

Imagen original: cameraman.pgm

Tasa: 0.387

Tasa 0.234 bpp

Tasa 0.126 bpp

Comparación de imágenes de JPEG2000, con respecto a nuestro *software*.
Imagen original:

JPEG2000

PSNR = 27.03 dB
Tasa aprox. de 0.20 bpp

Nuestro *software*

PSNR = 28.89 dB
Tasa de 0.15 bpp

JPEG2000

PSNR=25.9 dB
Tasa de 0.2343 bpp

Nuestro *software*

PSNR=19.89 dB
Tasa de 0.11 bpp

5.2 Conclusiones

En esta tesis ha sido estudiado, desarrollado e implementado un *software* de compresión de imágenes de niveles de grises, así como también un simulador de transmisión progresiva de las imágenes codificadas.

Para ser compatibles con la norma JPEG2000, las imágenes son transformadas utilizando las transformadas *wavelet* biortogonales LeGall 5/3 y Daubechies 9/7, posteriormente los coeficientes *wavelet* obtenidos de la transformación son ordenados de acuerdo a la energía, o información, que poseen mediante el algoritmo EZW, dejando lista la información para su posterior transmisión progresiva.

El *software* de compresión trabaja con imágenes cuadradas de escala de grises y consigue una buena compresión y calidad de las imágenes reconstruidas, tanto para la compresión con pérdidas como para la compresión sin pérdidas, como se puede ver en los resultados.

En seguida se enlistan las conclusiones correspondientes a cada tabla y/o gráfica de resultados.

Tabla 5.1

En esta grafica el costo de entropía va disminuyendo conforme el número de niveles de descomposición *wavelet* va aumentando. Esto es debido a que en cada nivel de descomposición se concentra mayor información en menos coeficientes, por lo tanto, considerar unos cuantos coeficientes de la banda paso bajas, puede descartar muchos coeficientes que se encuentren en un árbol de ceros, ahorrando muchas iteraciones.

El costo es mayor con la *wavelet* LeGall (5/3) ya que, debido a que es una *wavelet* reversible, contiene más información.

Tabla 5.2

Muestra que conforme el umbral disminuye, la razón PSNR aumenta, por lo que la calidad de la imagen es mejor. Esto es debido a que cuando el umbral es más chico se obtiene mayor información para su reconstrucción. También se puede observar que generalmente con la *wavelet* LeGall 5/3 se obtiene una mejor calidad que con la Daubechies 9/7, hecho normal dada la característica de reconstrucción perfecta de la primera.

Tabla 5.3

Con respecto a los datos anteriores se muestra que al ir aumentando los niveles de descomposición, la razón PSNR (calidad de la imagen) disminuye, además de que si el umbral es más chico se obtiene una mejor calidad de la imagen reconstruida.

Tabla 5.4

Se puede ver que cuando se alcanzan umbrales de compresión grandes, la tasa de compresión es también grande, pero la calidad de la imagen es baja, debido a la poca información que se utilizó en su reconstrucción.

Con respecto a los niveles de descomposición *wavelet*, se puede ver que al ir aumentando el número de niveles la tasa de compresión para cada umbral aumenta, y la razón PSNR disminuye.

Tabla 5.5

En la primera tabla se puede observar que el tiempo de ejecución mínimo es de 2 minutos con 20 segundos para la compresión, suponiendo que trabajamos con una imagen de tamaño 256x256 pixeles. Hay que considerar que estas ejecuciones realizaron la compresión hasta obtener la mayor información posible, es decir, hasta que el umbral alcanzó el valor 1.

Considerando que una reconstrucción aceptable es con un umbral de 16, se obtiene un tiempo de 7 segundos, lo cual es muy aceptable tomando en cuenta que tenemos una noción de la imagen desde un inicio y no tenemos que esperar los 7 segundos para apreciar el contenido.

Tabla 5.6

En esta tabla y en las imágenes mostradas se puede observar que, con una compresión de la imagen de hasta 5 veces, nuestro programa reconstruye con una mejor calidad que el estándar JPEG2000. Sin embargo con una compresión mayor, tiene mejor calidad el JPEG2000.

Es importante considerar que las imágenes que se consideran de buena calidad deben tener una razón PSNR de al menos 30 [dB] y para efectos prácticos aún es aceptable cuando se tienen 25 [dB] de PSNR. Con esta consideración puede resultar más eficiente el *software* que realizamos que el JPEG2000.

5.3 Futuros trabajos

Sabemos que el algoritmo EZW tiene cierto grado de detección y corrección de errores, sin embargo, nuestro *software* realiza la simulación de una transmisión sin considerar el ruido del canal que puede provocar errores en los datos afectando la información. Un trabajo posterior puede incluir todos estos efectos de la transmisión y analizar qué tan robusta es la técnica de compresión. Una vez simulada, se podría realizar una transmisión en un canal con ruido y manipular la tasa de compresión para adaptarla al ancho de banda.

Por otro lado, debido a que el objetivo de esta tesis es desarrollar un *software* que realizara la transmisión progresiva usando la transformada *wavelet*, únicamente fue enfocado a imágenes en blanco y negro, sin embargo un trabajo interesante sería modificar el *software* para poder aplicarlo a los diferentes canales de las imágenes a color y comprobar la eficiencia de la técnica. Esto puede ser llevado acabo incluso para el video.

Además de que en este trabajo solo se usaron las dos wavelets compatibles con el estándar JPEG2000:

- LeGall 5/3
- Daubechies 9/7.

Por lo que se podrían hacer pruebas con otras familias de *wavelets* y que de forma automática, dependiendo de la imagen, identificara la mejor wavelet para cada caso.

Una característica interesante de la compresión con JPEG2000 es el poder transmitir más información de una región de la imagen que consideremos nos interese más. Esto también puede ser implementado en nuestro algoritmo en algún trabajo futuro modificando alguno de sus módulos.

ANEXO Código de implementación en C ++.

El código con el cual se obtuvieron los resultados de ésta tesis esta conformado por varios archivos que poseen diversas funciones y librerías, los cuales son utilizados en el código principal.

El software implementado corre sobre Linux y el código principal se muestra a continuación.

```
//Librerias éstandar

#include <iostream.h>
#include <string.h>
#include <math.h>

//Librerias creadas

#include "vector.h"
#include "fdwt.h"
#include "EZW.hpp"

//Declaración de funciones

int psnr(Vector & original, Vector & copia);
int CostoEntropia(Vector & original ,Vector & copia);

//Cuerpo principal del programa

int main()
{

      // Se pide el nombre del archivo con el que se trabajara
      //(nombre de la imagen)

      cout << "Escribe el nombre de la imagen original "
           <<"(incluyendo la extensión): ";
      char cad[15];
      cin >> cad;
      char cad2[15];
      char *nombreOut;
      char *nombre;
      nombre=new char[strlen(cad)+1];
      strcpy(nombre,cad);


      //el archivo original se guarda como un vector en “Vector
      //original” y en “Vector v”

      Vector original=abre(nombre);
```

```
      Vector v=abre(nombre);

      //Se obtiene el tamaño de la imagen N x N

      const int N=int(pow(original.tam,.5));

      cout << "Tamaño de la imagen: " << N << endl;


// se indica por medio del valor de la variable sentido, que
//proceso se quiere realizar, ya sea “Compresión “ o
//“Descompresión”

      int sentido;
      cout << "Indica el proceso a realizar.\n\tCompresión (1) o
descompresión(-1): ";
      cin >> sentido;

//La variable T guardara el valor del umbral deseado en la
//descompresión del algoritmo EZW, en el caso de la compresión T
//tiene el valor de “cero”

      int T;
      if (sentido==-1)
      {
            cout << "Escribe un nombre para la imagen
descomprimida:";
            cin >> cad2;

            nombreOut=new char[strlen(cad2)+1];
            strcpy(nombreOut,cad2);

            cout << "Theshold mínimo: ";
            cin >> T;
      }
      else
            T=0;

//La variable NL, contendrá el número de niveles de descomposición
//Wavelet

      int NL;
      cout << "Niveles de descomposición: ";
      cin >> NL;

      int tam=N*N;
      int NLmax=0;
      for (int i=N; i>1; i/=2)

// Se cuenta cuántas veces el tamaño total de la imagen es
//divisible entre 2, para así obtener el número de niveles
//posibles

      NLmax++;

      if (NL>NLmax)
      {
```

```
            cout << "\nError. El maximo número de niveles es " <<
            NLmax << endl;
            exit(1);
      }


//la variable “w” contendrá el número correspondiente a la wavelet
//con la que se realizara el proceso de compresión y descompresión

      int w;
      cout  << "\n1 --> LeGall 5/3" << endl
            << "2 --> Daub   9/7" << endl
            << "Elija una wavelet: ";

      cin >> w;
      cout << endl;
      system("date");



/* ------------ TRANSFORMADA O ANÀLISIS   ----------------  */

if (sentido>0)
{
      cout << "\nNivel\n";

//”n” cuenta los niveles y “m” aumenta en potencia de 2 para
//dividir el tamaño de la matriz

      for (int n=1, m=1; n<=NL; n++, m*=2)
      {

//vector del tamaño de un renglón de las matrices LL, LH, HL, HH

            Vector aux(N/m);           // Renglones
            for (int ren=0; ren<N/m; ren++)
            {
                  // Obtiene un renglón del vector v

                  aux=v.obten(ren*N+1,1,ren*N+N/m);

                  // Reemplaza el renglón por su transformado

                  v=v.coloca(fdwt(aux,w),ren*N+1,1,ren*N+N/m);
                  }
            //columnas
            for (int col=1; col<=N/m; col++)
            {
// Obtiene una columna del resultado anterior

                  aux=v.obten(col,N,N*N/m);

// Remplaza la columna por su transformada

                  v=v.coloca(fdwt(aux,w),col,N,N*N/m);
            }
            cout << n << "...\t";
```

```
//Se obtiene el costo de entropía

            CostoEntropia(original,v);
      }
      cout << "transformado.\n";

//Se realiza el algoritmo EZW

      EZW(v.vect,N,T);

}



/* --------------- ANTITRANSFORMADA O SÍNTESIS ---------------  */

else if (sentido < 0)
{
      double *vec;
      Vector vi(N*N);
//Se irán sumando los resultados en la matriz vi_tot

      Vector vi_tot(N*N);
      char c=65;
      int t=512;

   while (T<t)
   {
      // se realiza el UNEZW

      vec=UNEZW(t, N, c);
      Vector vi(vec, N*N);

      vi_tot=vi_tot.masAbs(vi);
      vi=vi_tot;
      cout << "Nivel\t";

      for (int n=1, m=int(pow(2,NL-1)); n<=NL; n++, m/=2)
      {
            Vector aux(N/m);
            // Columnas
            for (int col=1; col<=N/m; col++)
            {
                  aux=vi.obten(col,N,N*N/m);
                  vi=vi.coloca(ifdwt(aux,w),col,N,N*N/m);
            }

            // Renglones
            for (int ren=0; ren<N/m; ren++)
            {
                  aux=vi.obten(ren*N+1,1,ren*N+N/m);
                  vi=vi.coloca(ifdwt(aux,w),ren*N+1,1,ren*N+N/m);
            }

            cout << NL-n+1 << "...\t";
```

```
      }

      cout << "antitransformado\n";
//Se normaliza el valor de los coeficientes
      vi.normaliza();
      escribeImagen(nombreOut,vi);
//Se obtiene el PSNR de la imagen reconstruida
      psnr(original,vi);

      c++;

   }
}
else
      cout << "Error. El sentido de la transformada no está
definido correctamente" << endl;

      system("date");
      cout << endl;

      return 0;
}



/* --------------- Función PSNR --------------------- */

int psnr(Vector & original, Vector & copia)
{
      double suma = 0;

      for (int i=0; i<original.tam; i++)
            suma = suma + pow((original.vect[i]-copia.vect[i]),2);

      if (suma==0)
      {
            cout << "PSNR = Infinito\n";
            return 0;
      }

      double MSE=suma/original.tam;
      double RMSE = pow(MSE,0.5);
      double PSNR = 20*log10(255/RMSE);
      cout << "PSNR = " << PSNR << endl;
      return 0;

} //Cierra función psnr



/* --------------- Función costo de entropía ------------------*/

int CostoEntropia(Vector & original ,Vector & copia)
{
 double EnergiaOriginal=0;
//para obtener la energia original de la imagen
 for(int e=0;e<original.tam;e++)
      {
```

```
      EnergiaOriginal+=original.vect[e]*original.vect[e];
     }

//para el costo
  double costo=0;
     for(int i=0;i<original.tam; i++)
     {
          if(copia.vect[i]!=0)
          {
     costo+=((copia.vect[i]*copia.vect[i])/EnergiaOriginal)*(log((
     copia.vect[i]*copia.vect[i])/EnergiaOriginal));

          }//fin if

     }

cout  << "costo= \t" << -1*costo << endl;

return 0;

}//cierra funcion costo entropia
```

Para correr el código anterior se necesita teclear los siguientes comandos en el shell de linux:

1. g++ -c main.cpp vector.cpp fdwt.cpp ezwdos.cpp unezwdos.cpp matrix2d.cpp list.cpp fifo.cpp abrearchivo.cpp -Wno-deprecated
2. g++ -o mainTx.exe vector.o fdwt.o ezwdos.o unezwdos.o matrix2d.o list.cpp fifo.o abrearchivo.o main.o -Wno-deprecated
3. mainTx.exe

Bibliografía y referencias

[1] Maria Petrou and Panagiota bosdogianni, Image Processing The Fundamentals, John Wiley & sons, LTD.

Clasifcacion:TA1637 P48

[2] Stanford Goldman, “Information Theory”, Dover Publications, Inc. Nueva York, 1968.

[3] A Murat, “Digital Video Processing”, Cherry C., On human Communication. MIT Press, U.S.A., 1966.

TK6680.5 T45

[4] Gordon Raisbeck, Information Theory, an introduction for scientists and Engineers,The M.I.Press.

clasificacion : Q365 R3

[5] F. Elliot,K.Ramamohan Rao ,Fast Transforms Algorithms Analyses,Applications,Douglas., Academic Press , INC.

Clasificaciòn: QA403.5 E54 EJ.1

[6] Y. Meyer, *Wavelets: Algorithms and Applications.* Philadelphia: SIAM, 1993

[7] S. Mallat “A theory of multiresolution signal decomposition: the wavelet representation”, *IEEE Trans. Pattern Anal. Machina Intell.* 1989

[8] Ingrid Daubechies, “Ten lectures on wavelets”, Rutgers University and AT&T Bell Laboratorios

.

[9] David S. Taubman and Michael W. Marcellin, Fellow, IEEE, “JPEG 2000: Standard fpr Interactive Imaging”,vol 90, No 8,pp. 1336-1357, August ,2002.

[10] Athanassions Skodras, Charilaos Christopoulos, and Touradj Ebrahimi, “The JPEG 2000 Still Image Compression Standard”, IEEE Signal Processing Magazine, pp 36-58, September, 2001.

[11] R. Montúfar-Chaveznava, F. García-Ugalde and B. Psenicka, “Quantization Effects in Pyramidal Adaptive Approximation Image Coding Techniques”.

[12] Jerome M. Shapiro, “Embedded Image Coding Using Zerotrees of Wavelet Coefficients”, Vol. 41, No 12, pp 3445-3462,December, 1993.

[13] Ismail Avcibas, Bülent Sankur, Khalis Sayood, “Statistical evaluation of imege quality measures“, Vol 11, No 2, pp 206-223, April, 2002.

[14] Michael W. Marcellin, Michael J. Gormish, Ali Bilgin, Martin P. Boliek, “An Overview of JPEG-2000”, pp 523-541,2000.

[15] Lorenzo Peotta, Lorenzo Granai, Pierre Vandergheynst, “Image compression using an edge adapted redundant dictionary and wavelets”, May, 2005.

[16] José Manuel Alarcón Roldán “Compresión de imágenes y vídeo mediante la transformada wavelet”

[17] Nieves Baena Martín “El futuro de la multimedia. Técnicas de compresión de la información digital. Transformada de wavelet. Aplicaciones”. http://www.ieev.uma.es/edutec97/edu97_c3/2-3-10.htm

[18] “Algoritmos de compresión de imágenes sin movimiento para comunicaciones móviles (3G) utilizando teoría de wavelets Teoría de Wavelets” Capítulo 2 http://catarina.udlap.mx:9090/u_dl_a/tales/documentos/lem/perez_r_oa/capitulo2.pdf

[19] “A wavelet tour of signal processing by stéphane mallat”. Wavelet Basis. http://cas.ensmp.fr/~chaplais/Wavetour_presentation/ondelettes/Wavelets.html

[20] ROBI POLIKAR “The Wavelet Tutorial”. http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html

[21] C. Valens, 1999-2004 “A Really Friendly Guide to Wavelets” http://perso.orange.fr/polyvalens/clemens/wavelets/wavelets.html

[22] Lorenzo Peotta, Lorenzo Granai, Pierre Vandergheynst “Image compression using an edge adapted redundant dictionary and wavelets” http://lts1pc19.epfl.ch/repository/Peotta2004_1152.pdf