Universidad Nacional Autónoma de México

FACULTAD DE INGENIERÍA

"Estudio y simulación de un método de compresión de imágenes digitales por medio de una transformada wavelet y de una estructura de árbol de ceros"

T E S I S

que para obtener el título de:
Ingeniero Eléctrico Electrónico
Módulo: Electrónica para Comunicaciones

PRESENTA

Iris Janett Campos Carcaño

Director de Tesis: Dr. Francisco García Ugalde

Ciudad Universitaria 2003

Índice

Título **Página**

Prefacio

El objetivo fundamental de esta tesis es el de desarrollar un sistema de compresión de imágenes digitales basado principalmente en el algoritmo conocido con las siglas EZW (*embedded zerotree wavelet*) como una alternativa a los métodos de compresión convencionales.

Este documento se conforma de la siguiente manera, en el Capitulo 1 se muestra una explicación general del contexto, el alcance, la justificación, la relevancia del trabajo realizado, el método y la metodología para desarrollar la tesis, este capítulo tiene como fin principal que el lector comprenda de una forma rápida el problema y la solución implementada, es decir, es un resumen del estudio hecho.

En el Capitulo 2 se explican los conceptos y la teoría de la compresión de imágenes, que son la base fundamental de esta tesis.

El Capitulo 3 pretende demostrar qué es una función *wavelet*, qué es una transformada *wavelet*, el concepto de descomposición multiresolución, así como los tipos de transformada *wavelet* existentes. La parte teórica de estos conceptos es demasiado extensa, por lo que resultaría muy difícil describirla en detalle en este trabajo, pero la aplicación de esta transformación es uno de los puntos importantes en el método de compresión y es por eso qué se describe brevemente en este capitulo.

En el Capitulo 4 se describe con profundidad al algoritmo EZW, sus características fundamentales como por ejemplo, la naturaleza progresiva de su salida, lo cual es muy útil cuando el ancho de banda disponible para la transmisión es variable. Se describe detalladamente el funcionamiento del algoritmo y los pasos que lo conforman. En este capitulo podemos ver las simulaciones de la solución implementada, además de la comparación de los resultados obtenidos con los correspondientes a la norma JPEG.

El Capitulo 5 muestra las conclusiones y los posibles proyectos que pueden desarrollarse a partir de esta tesis.

Por último se proporcionan las referencias del material bibliográfico utilizado.

CAPITULO 1

Este capítulo tiene como objetivo el mostrar la idea y los resultados generales del trabajo desarrollado en la presente tesis. De esta manera el lector que lo lea, podrá consultar de una forma rápida el problema planteado y si desea profundizar más en el tema podrá leer los capítulos subsecuentes.

1.1 Definición General del problema

Actualmente la compresión de imágenes forma parte importante en el área de investigación del procesamiento digital de imágenes, ya que la cantidad de datos para crear una imagen que se obtiene desde un dispositivo que convierte una señal analógica en digital es muy grande, así como el número de imágenes [1]. También es fundamental en el área de transmisión y recepción de datos a través de una red de computadoras, por ejemplo Internet, o una red de telecomunicaciones, para la transmisión de señal digital de tipo televisión (HDTV).

En el caso de Internet, es fácil percibir cuando una página contiene algunas o muchas imágenes, ya que regularmente la página tarda algunos momentos en desplegar su contenido total. Aunque en muchas ocasiones el retraso al desplegarla, pueda ser causa de las imágenes, también dependerá mucho de varios factores que componen la red, como su velocidad y ancho de banda.

En un sistema de recepción y transmisión de datos vía satélite, la importancia que tiene la compresión de imágenes, es la de reducir la cantidad de datos para representar las imágenes, con el propósito de que los datos a transmitir sean el menor número posible, y con la garantía de que cuando se quieran reconstruir, se vean tal y como son originalmente.

La compresión de datos tiene una aplicación importante en las áreas de transmisión y almacenamiento de datos y actualmente es crucial para el crecimiento de la informática multimedia. Muchas de las aplicaciones para su procesamiento requieren almacenamiento de grandes volúmenes de datos y el número de tales aplicaciones se incrementa constantemente con el uso de las computadoras. Al mismo tiempo la proliferación de las comunicaciones en las redes de computadoras, hace que la transferencia masiva de datos sobre los canales de comunicación sea más demandante, aquí es donde puede hacer su aparición la transmisión progresiva. La compresión de datos, para ser almacenados

o transmitidos, reduce costos de almacenamiento o comunicación; ya que cuando la cantidad de datos a ser transmitidos se reduce, se incrementa proporcionalmente la capacidad de los canales de comunicaciones.

Por todo esto, 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 el aumento constante de cantidad de información a enviar o almacenar.

En consecuencia en este trabajo de tesis se explora una técnica de compresión de imágenes diferente a los estándares conocidos, para lo cual se implementará un algoritmo de codificación y se evaluará su eficiencia y su robustez. Dicho algoritmo es conocido por sus siglas como EZW (*embedded zerotree wavelet*). El EZW utiliza las técnicas matemáticas recientes del análisis de *wavelets*, que representan un marco de trabajo alterno a lo que ha sido por los últimos dos siglos el estado del arte en el análisis de señales: el Análisis de Fourier. Este algoritmo de codificación tiene su base en el aprovechamiento de una de las propiedades que tienen las transformadas *wavelets*: la estructura de descomposición multiresolución, en la cual los coeficientes producidos por transformada presentan cierta estructura que se puede comparar con la estructura de un árbol. En este árbol, cuando un nodo llamado "padre" es de valor bajo, es muy probable que sus nodos llamados "hijos" también sean de bajo valor. Por lo que el algoritmo explota está propiedad para propósitos de compresión.

Gracias a esta estructura, después de que el algoritmo EZW procesa los coeficientes, la transmisión puede ser progresiva y de varios niveles de tasa de transmisión, medida en bits/seg. Ya que una vez estructurados los coeficientes en forma de árbol, se pueden enviar primero los más significativos. Entonces se tiene la libertad de escoger la tasa de transmisión que se desea, siempre procurando mantener la mejor relación 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.

Para realizar la compresión digital de imágenes es necesario estudiar: el principio de la compresión de imágenes, las técnicas basadas en transformadas y las propiedades de la transformada *wavelet* . La siguiente etapa consiste en el estudio de la codificación de los datos generados por el análisis y finalmente en evaluar la eficiencia de este método de análisis.

Los puntos antes citados se desarrollan en este trabajo de tesis de manera que se demuestra la factibilidad de la utilización de este algoritmo en la compresión y la transmisión de imágenes.

1.2 Objetivos

Los objetivos principales planteados en este trabajo son:

- Entender claramente los diferentes métodos de compresión de imágenes digitales, así como las operaciones básicas que los componen.
- Clasificar los métodos de compresión de imágenes y escoger los que tengan las mejores propiedades desde el punto de vista de la tasa de compresión y de la razón señal a ruido.
- Estudiar y evaluar el desempeño de un algoritmo de compresión de imágenes digitales denominado EZW (*embedded zerotree wavelet*), que explota la estructura de una transformada *wavelet* para obtener una codificación más adecuada para la compresión y la transmisión progresiva.
- Hacer simulaciones y evaluaciones del codificador diseñado con el algoritmo estudiado, para poder comparar sus resultados con los obtenidos en la norma JPEG.

Con el cumplimiento de estos objetivos se logra conocer y proponer nuevas técnicas de compresión, que pueden ser iguales o más eficientes que los estándares existentes

1.3 El Algoritmo EZW

El método de compresión EZW fue propuesto por Shapiro en 1993. Las principales características del algoritmo EZW incluyen una representación multiresolución compacta de las imágenes por medio de la transformada *wavelet* discreta; la codificación, con la utilización de la estructura *zerotree*, de los coeficientes *wavelet* significativos lo cual provee mapas de significancia, que son mapas binarios que indican la posición de los coeficientes significativos , desde el punto de vista de su contenido energético; otra característica es la cuantización por aproximaciones sucesivas de los coeficientes, lo cual facilita que el algoritmo produzca una salida llamada embebida; el uso de un codificador aritmético adaptativo, que es un método de codificación de entropía muy eficiente para codificar cadenas de bits y por último, el algoritmo corre de manera secuencial y termina hasta alcanzar una tasa de bits requerida.

El funcionamiento del algoritmo es sencillo; después de haber aplicado la transformada *wavelet* discreta a la imagen original, la codificación es entonces implementada basada en una prueba de significancia con respecto a un umbral dado. El *zerotree* (T) se define basándose en el hecho de que si un coeficiente

wavelet localizado en una escala alta es insignificante, entonces sus descendientes, por ejemplo, todos los coeficientes *wavelet* de la misma orientación y en la misma localización espacial en escalas más bajas, serán muy probablemente insignificantes.

Además de T, se definen otros tres símbolos que son usados en el algoritmo: (Z) cero aislado, (P) significativo positivo y (N) significativo negativo.

Conforme el umbral decrece, se encontrarán más coeficientes significativos, lo cual dará una mejor aproximación de la imagen original. En otras palabras, es un esquema de codificación llamada embebida.

Considerando esta estructura, en este estudio se implementaron las siguientes etapas:

- Una transformada *wavelet* discreta
- Para un umbral dado, se aplica un paso llamado dominante para obtener los símbolos (T, Z, N y P) y también se aplica otro paso llamado subordinado para refinar los niveles de reconstrucción de los coeficientes significativos.
- Una codificación aritmética adaptativa la cual comprime la cadena de bits, y que cuenta sólo con cuatro símbolos, producida por los pasos mencionados.
- Una transformada *wavelet* discreta inversa.
- Se reduce el umbral a la mitad para crear una nueva cadena de bits, con lo cual se obtiene una mejor aproximación que la anterior.

Como se puede observar el desempeño de este algoritmo se alcanza con una técnica que no requiere de ningún entrenamiento, ni de tablas previamente almacenadas, además de que no requiere de un conocimiento previo de la imagen original, por lo que funciona para cualquier tipo de imágenes.

En esta tesis, este algoritmo se implementó en lenguaje C, para lo cual la base principal para esta implementación fue el trabajo escrito por Jerome M. Shapiro que fue publicado en 1993, tomando en cuenta una técnica para la identificación de un *zerotree* que fue publicada por él mismo en 1996. En el capítulo 4 se puede observar que el desempeño de este algoritmo es competitivo con técnicas normativas conocidas como JPEG.

CAPÍTULO 2

Con los años, la necesidad de comprimir las imágenes ha crecido constantemente, ya que un número creciente aplicaciones dependen de una eficaz manipulación. Almacenamiento y transmisión de imágenes.

De igual manera, este trabajo de investigación tiene su bases teóricas en los conceptos de la compresión de imágenes.

Por lo tanto en este capítulo, se revisarán brevemente los fundamentos de la compresión de imágenes. En un principio, se introducirá a los conceptos que colectivamente forman la teoría de esta disciplina (tipos de redundancias, teoría de la información etc.). También se describen los esquemas de compresión así como las técnicas usadas en su procesamiento. Terminando con una explicación más profunda del tipo de codificación utilizado en este trabajo (codificación por transformación), así como los criterios de calidad utilizados para evaluar una imagen después de un proceso de codificación-decodificación.

2.1 La Imagen Digital

La palabra imagen tiene muchos significados, para definirla de una manera simple podemos decir que una imagen es la representación de una persona, animal o cosa. En general una imagen natural puede ser representada mediante una función bidimensional continua, en la cual el nivel de intensidad o los diferentes tonos de gris que presenta, son funciones de la posición.Técnicamente hablando una imagen es una función bidimensional de intensidad de luz f(x,y) donde x y y representan las coordenadas espaciales y el valor de f en un punto cualquiera (x,y) es proporcional a la intensidad (nivel de gris) de la imagen en ese punto.

Una imagen que se ha discretizado es una imagen digital. Una imagen digital es una matriz cuyos índices de fila y columna identifican un punto en la imagen y el valor del correspondiente elemento de dicha matriz, comúnmente denominado pixel, nos indica el nivel de gris en ese punto. Cada pixel se representa mediante un número real, o un conjunto de números reales en un limitado número de bits. Basándonos en la exactitud de su representación , las imágenes se clasifican en tres categorías, imágenes blanco y negro, imágenes en escala de grises e imágenes a color. Las características de cada una son :

- En las imágenes blanco y negro cada pixel está representado por un bit. A este tipo de imágenes también se les llama imágenes bi-nivel, binarias o bi-tonales.

- En las imágenes en escala de gris, cada pixel está representado por un nivel de luminancia (o intensidad). En las imágenes pictóricas las escalas de gris están representadas por 256 niveles de gris, u ocho bits.
- Las imágenes a color tienen múltiples componentes. Cada pixel, en este tipo de imágenes, puede ser representado por los componentes de luminancia y crominancia. Por ejemplo en el sistema de transmisión de NTSC (*National Television Systems Committee*), se usa la luminancia Y, y dos componentes de crominancia I y Q para representar el color. Los estándares MPEG y CCITT usan una luminancia Y y dos crominancias C_B y C_R para la representación del color. [2]

La demanda para el manejo de imágenes en forma digital se ha incrementado aceleradamente durante los últimos años. Gracias al mejoramiento del desempeño y a la significativa reducción en los costos de los *scanners*, cámaras digitales, impresoras y otros medios digitales, la imagen puede ser convertida a formato digital. Muchas modalidades de imágenes en medicina, como una tomografía computarizada o imágenes de resonancia magnética, generan imágenes directamente en forma digital.
El uso de gráficos por computadora en las ramas de publicidad y entretenimiento está extendiéndose y su uso en la visualización científica y en las aplicaciones de ingeniería, continúa creciendo a pasos agigantados. La razón por el interés en imágenes digitales es clara: la representación de imágenes digitales permite que la información visual sea fácilmente manipulada por la computadora en formas útiles y novedosas. Este hecho, combinado con el crecimiento exponencial de la capacidad de cómputo durante las últimas décadas, ha promovido el uso de sistemas de imágenes en diversos campos como: astronomía, percepción remota, medicina, foto periodismo, artes gráficas, publicidad, manufactura, etc.

Sin embargo, existe un problema con este tipo de imágenes, y es la gran cantidad de bits que se requieren para almacenarlas. Afortunadamente las imágenes digitales contienen una gran cantidad de redundancia. La compresión de imágenes, intenta aprovechar esta redundancia para reducir el número de bits requeridos para representar la imagen. Esto puede producir un ahorro significativo en la memoria requerida para el almacenamiento, o en la capacidad del canal que se requiere para la transmisión de la imagen.

2.2 Fundamentos de la compresión de imágenes

Como ya se mencionó, cuando se crea una imagen digital, se tiene como resultado una enorme cantidad de datos; esta cantidad puede ser tan grande que su almacenamiento, procesamiento, y comunicación pueden llegar a ser desmesurados.

En la tabla siguiente se muestra el espacio en disco, el ancho de banda , y el tiempo de transmisión necesario para el almacenamiento y la transmisión de algunos ejemplos de datos (imagen, audio y video) sin comprimir.

Datos Multimedia	**Tamaño o Duración**	**Bits/Píxel o Bits/Muestra**	**Tamaño sin compresión (B = bytes)**	**Ancho de Banda de transmisión (b = bits)**	**Tiempo de transmisión (usando un Módem de 28.8K)**
Página de texto	11" x 8.5"	Resolución Variable	4-8 KB	32-64 Kb/page	1.1 - 2.2 seg
Voz con calidad telefónica	10 sec	8 bps	80 KB	64 Kb/sec	22.2 seg
Imagen en escala de grises	512 x 512	8 bpp	262 KB	2.1 Mb/image	1 min 13 seg
Imagen a color	512 x 512	24 bpp	786 KB	6.29 Mb/image	3 min 39 seg
Imagen médica	2048 x 1680	12 bpp	5.16 MB	41.3 Mb/image	23 min 54 seg
Video	640 x 480, 1 min (30 frames/sec)	24 bpp	1.66 GB	221 Mb/sec	5 días 8 hrs

Tabla 2.1. Ejemplos de datos multimedia sin compresión y el espacio necesario para su almacenamiento.

Debido a que una imagen es un conjunto de información que generalmente tiene una extensión considerable en la memoria de una computadora, en un inicio, se requería una gran capacidad de memoria de almacenamiento para conservar las imágenes en formato digital. Aunado a esto, la evolución de las redes de comunicaciones ha hecho que la transmisión de imágenes sea muy común, y si transmitimos una imagen sin procesar es necesario un gran ancho de banda para que la transmisión se realice en un tiempo relativamente pequeño.

En el almacenamiento y manipulación de imágenes digitales en general, los métodos de compresión juegan un importante papel al hacer posible que éstas

sean almacenadas en menor espacio, sin provocar una notable degradación de la información que contienen.

El término compresión de datos se refiere al proceso de reducción del volumen de datos necesarios para representar una determinada cantidad de información. Hay que saber distinguir entre el significado de datos y el significado de información, ya que no son sinónimos Los datos son los medios a través de los que se transporta la información. Se pueden utilizar distintas cantidades de datos para describir la misma cantidad de información. Por lo tanto, hay datos que proporcionan información sin relevancia. Esto es lo que se conoce como redundancia de los datos [3] . La redundancia de los datos es un punto clave en la compresión de datos digitales. La redundancia presente en una imagen digital es altamente dependiente del sistema que se use para formar la imagen así como de los parámetros usados para representarla.

Fácilmente se puede observar que la mayoría de las imágenes digitalizadas contienen gran cantidad de redundancia, es decir, los valores de los tonos de gris en algunas zonas de las imágenes están correlacionados, o dicho de otra manera, muchos valores de tonos de gris se pueden predecir a partir de los valores de sus puntos vecinos.

La idea fundamental de la compresión de imágenes es entonces tratar de representar dichas imágenes mediante un conjunto de datos no correlacionados entre sí. De manera que cada uno de estos datos nos proporcione información sobre alguna propiedad única de la imagen, única en el sentido de que esa propiedad no se puede predecir a partir de los otros datos. Es por eso que debido a la redundancia, el número real de bits para representar una imagen puede ser substancialmente menor.

En el caso de las imágenes, existen tres maneras de reducir el número de datos redundantes [3] :

eliminar código redundante

eliminar píxeles redundantes

eliminar redundancia visual.

Redundancia de codificación

El código de una imagen representa el cuerpo de la información mediante un conjunto de símbolos. La eliminación del código redundante consiste en utilizar el menor número de símbolos para representar la información.

Las técnicas de compresión por codificación de Huffman y codificación aritmética utilizan cálculos estadísticos para lograr eliminar este tipo de redundancia y reducir la expansión original de los datos.

Redundancia entre pixeles

La mayoría de las imágenes presentan semejanzas o correlaciones entre sus píxeles vecinos. Estas correlaciones se deben a la existencia de estructuras similares en las imágenes, puesto que no son completamente aleatorias. De esta manera, el valor de un píxel puede emplearse para predecir el de sus vecinos.

Las técnicas de compresión Lempel-Ziv implementan algoritmos basados en sustituciones para lograr la eliminación de esta redundancia.

Redundancia Visual

El ojo humano responde con diferente sensibilidad a la información visual que recibe. La información a la que es menos sensible se puede descartar sin afectar a la percepción de la imagen. Se disminuye así lo que se conoce como redundancia visual.

Las tres técnicas mencionadas para la reducción de las redundancias permiten comprimir la cantidad de datos necesarios para representar una imagen. Sin embargo, estas técnicas se combinan para construir sistemas prácticos de compresión de imágenes.

Si aplicamos a una imagen una técnica de compresión, podría suponerse que aplicando la técnica varias veces conseguiríamos comprimir más la imagen. Desgraciadamente esto no ocurre porque la operación de compresión elimina una forma particular de redundancia, y una vez eliminada, no puede ser otra vez eliminada. Normalmente, cuando una operación de compresión se aplica iterativamente a una imagen, el tamaño de la imagen comprimida alcanza un límite a partir del cual ya no se reduce, o incluso puede aumentar.
A veces se consigue reducir más el tamaño de una imagen aplicando diferentes técnicas de compresión, cada una de las cuales elimina una forma diferente de redundancia. La reducción del tamaño de una imagen tiene un límite, marcado por las características intrínsecas de la redundancia, y por la calidad requerida en la imagen cuando se descomprime posteriormente.

Mediante la operación de compresión, el conjunto de datos de la imagen original queda reducido, en un conjunto de datos comprimidos. La descompresión convierte los datos comprimidos de nuevo a su forma original, o algo similar

Compresión y descompresión son operaciones llamadas de **codificación de imágenes**, debido a que hacen uso de métodos de codificación de datos, para representar una imagen de una forma más concisa.

Codificación significa un cambio de representación de la información que contiene una imagen con el objetivo de hacer explícitas ciertas entidades y entenderlas mejor. Dicho de otra forma, se sabe que, la información se representa por medio de datos que generalmente en los sistemas de despliegue visual se presentan como un conjunto finito de niveles de intensidad. Dicha representación, depende de la aplicación que se requiere, por ejemplo si lo que se desea es visualizar la información en una computadora, es la representación numérica espacial la que da idea de los diferentes niveles de intensidad; pero si deseamos conocer sus componentes frecuenciales, entonces por medio de la transformada de Fourier lo podemos hacer; si lo deseable es realizar una representación con el menor número de datos, entonces las técnicas de compresión son muy útiles.

2.3 Resumen de la teoría de la información.

Los principios de la compresión digital de imágenes se basan en la Teoría de la Información. Esta teoría permite analizar el contenido de información de una fuente de datos. Describe la posibilidad de representar determinada cantidad de información de una forma compacta, es decir, analiza el contenido de la información de una cantidad de datos, y establece cual es la cantidad mínima de bits, que puede ser usada para representar esa información. De esta forma, esta teoría indica que basta retener esta cantidad mínima de datos, la restante puede ser quitada porque es considerada redundante, de tal manera que el contenido de información no sufra pérdidas.

En 1948, Claude Shannon, padre de esta teoría, introdujo la medida de la información basada en una definición de probabilidad. Esta medida es la piedra fundamental de la teoría de las comunicaciones. La premisa fundamental de la teoría de la información es que puede modelarse como un fenómeno probabilístico que se puede medir de forma acorde con la intuición. Sea x_j un evento que ocurre con probabilidad $p(x_j)$. Si se informa que ha ocurrido el evento x_j , se dice que se han recibido:

$$I(x_j) = \log_a \frac{1}{p(x_j)} = -\log_a p(x_j)$$

unidades de información. La base de logaritmos utilizada es enteramente arbitraria y determina las unidades con que se mide la información. Se usan comúnmente los logaritmos de base 2 para la medida de la información, ya que la unidad de información es el bit. En la computadora podemos asociar cada 0 lógico y cada 1

lógico a un bit de información, suponiendo que cada uno de estos estados es igualmente probable.

La información promedio asociada con el resultado de un experimento, es más interesante que la información asociada con cada evento en particular. A la información promedio asociada con una variable aleatoria discreta, X, se le define como la entropía H (X), y se puede calcular como:

$$H(X) = E\{I(x_j)\} = -\sum_{j=1}^{n} p(x_j)\log_2 p(x_j)$$

donde n es el número de resultados posibles.

En el proceso por producir una teoría de las comunicaciones que pudiera ser aplicada por los ingenieros eléctricos para diseñar mejores sistemas de telecomunicaciones, Shannon definió una medida de entropía. La entropía puede ser considerada como la incertidumbre promedio, por lo que debe ser máxima cuando cada resultado sea igualmente posible. Supongamos que un proceso aleatorio tiene *n* posibles resultados y que p_n es una variable dependiente de otras probabilidades. Entonces:

$$p_n = 1 - (p_1 + p_2 + ...p_k + ... + p_{n-1})$$

donde p_j es la notación concisa de $p(x_j)$. La entropía asociada con el proceso es:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Por ejemplo, para una imagen con 256 niveles de gris, p_i es la probabilidad de cada uno de los *pixeles,* los cuales pueden tomar valores de 0 a 255. Es decir:

$$H = -\sum_{0}^{255} p_i \log_2 p_i$$

El cálculo de la entropía nos permitirá conocer la cantidad de información que se conserva en la imagen una vez transformada.

La entropía es una medida del contenido en información de la imagen. Si la entropía es elevada, la información contenida en la imagen tiende a ser muy impredecible. Es decir, una imagen con alta entropía contiene mucha aleatoriedad y poca redundancia. Si la entropía es baja, la información de la imagen es más predecible, contiene poca aleatoriedad y mucha redundancia. [4] [5]

2.4 Panorama de los métodos de compresión

Existen dos maneras de clasificar las técnicas de compresión de imágenes:

(a) **Compresión con pérdida vs. sin pérdida**: en la compresión sin pérdidas (*lossless)*, la imagen reconstruida, después de la compresión, es numéricamente idéntica a la imagen original. Estos métodos se caracterizan porque la tasa de compresión que proporcionan está limitada por la entropía (redundancia de datos) de la señal original. Entre estas técnicas destacan las que emplean métodos estadísticos, basados en la teoría de Shannon, que permite la compresión sin pérdida. Por ejemplo: codificación de Huffman, codificación aritmética y Lempel-Ziv, son métodos idóneos para la compresión de archivos; de igual manera este tipo de métodos pueden utilizarse en algunos campos de aplicación de las imágenes, como es el caso del campo de la medicina, en donde en algunos países es legalmente necesario el archivo de imágenes digitales no alteradas. Una imagen reconstruida utilizando los métodos de compresión con pérdidas (*lossy*) contiene cierta degradación con respecto a la original. Los métodos de compresión con pérdida de información logran alcanzar unas tasas de compresión más elevadas a costa de sufrir una pérdida de información sobre la imagen original. Por ejemplo: JPEG, compresión fractal, EZW, etc.

(b) **Compresión predictiva vs compresión por transformación**: En la codificación predictiva se elimina la redundancia que existe en el dominio espacio-temporal en el que se representan las muestras. Se realiza prediciendo el valor de una muestra a través de su vecindad espacial y/o temporal, es decir, se usa la información disponible, o la que ya se ha mandado para predecir valores futuros, y se codifica sólo la diferencia entre los valores predichos y los valores actuales. El rango de las diferencias de las muestras es normalmente menor que el rango de las muestras individuales, y por lo tanto se necesitará un menor número de bits para codificar dicho rango. Ya que esto es hecho en la imagen o en el dominio espacial, es relativamente fácil de implementar y se adapta fácilmente a las características de la imagen: un claro ejemplo de este tipo de compresión es el método DPCM (*Differential Pulse Code Modulation*). Por otra parte, en la compresión por transformación, se utiliza una transformada lineal, reversible para hacer corresponder la imagen con un conjunto de coeficientes de la trasformada, que después se cuantifican y se codifican. En la mayor parte

de las imágenes naturales, un número significativo de coeficientes tienen pequeñas magnitudes

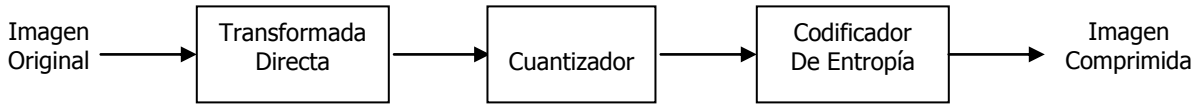
2.5 Codificación por transformación.

En este esquema de codificación se utiliza regularmente una transformada lineal reversible, para hacer corresponder a la imagen con un conjunto de coeficientes generados por la transformada. Estos coeficientes son los que se cuantifican y codifican.

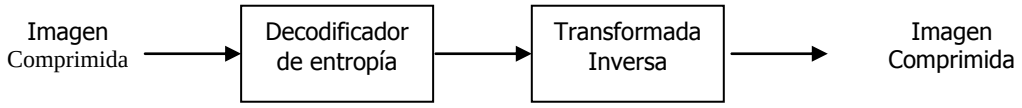
En este trabajo se utilizó la compresión basada en una transformación. Los inconvenientes de codificar imágenes empleando métodos por transformada han sido principalmente, la gran cantidad de cálculos necesarios. Sin embargo, recientemente se han introducido algoritmos rápidos que permiten efectuar tales cálculos más eficientemente.

Se han construido y estudiado extensivamente los sistemas de codificación por transformación basados en diversas transformadas, siendo las más comunes: la transformada de Fourier Discreta (DFT), coseno discreta (DCT), la transformada de Karhunen-Loeve (KLT), de Walsh-Hadamard (WHT), la transformada *wavelet* discreta (DWT), entre muchas otras. La elección de una determinada transformación en una aplicación concreta depende de la cantidad tolerable de errores de reconstrucción y de los recursos de cálculo disponibles. La compresión se consigue durante la cuantificación de los coeficientes de la transformada y no durante la etapa de transformación. Durante está, se logra que la energía total se distribuya solamente en algunos coeficientes significativos.

Un sistema típico usado en este tipo de compresión se muestra en la figura:



(a)



(b)

Figura 2.1. Sistema de codificación por transformación : (a) codificador; (b) decodificador

Como podemos ver el codificador consta de tres bloques llamados: (a)Transformador (b) Cuantizador y (c) Codificador de entropía. La compresión se logra aplicando una transformada linear para deshacer la correlación de los pixels, cuantificando los coeficientes que resultaron de la transformación y codificando los valores cuantificados.

Es importante destacar que un cuantizador y un codificador diseñados apropiadamente nos pueden ayudar a obtener una mejor compresión.

El decodificador implementa la secuencia inversa de las etapas (exceptuando la función de cuantificación) del codificador.

2.5.1 Transformador

En este bloque, la imagen es transformada de un dominio espacio/tiempo a algún nuevo dominio donde las redundancias entre pixeles son reducidas. A través de los años, se han desarrollado una gran variedad de transformadas entre ellas se tienen : la transformada discreta de Fourier (DFT), la transformada coseno discreta (DCT); la transformada de Walsh-Hadamard; y cuando usamos el algoritmo EZW, en este bloque se debe usar la transformada *Wavelet* (la transformada *wavelet* discreta, es la transformada *wavelet* más usada en la actualidad) que transforma la imagen original al dominio escala/tiempo.

Desde el punto de vista de la compresión, el éxito de una transformada es reducir la correlación de los pixeles de la imagen, comúnmente se busca que la energía de la señal sea empaquetada o distribuida en el menor número de coeficientes posible [6].

Cuando una transformación es separable se realiza en una dimensión (1-D), ya sea sobre las columnas o los renglones de la imagen y cuando se quiere hacer la transformada en dos dimensiones (2-D), se realiza separadamente tanto en las columnas como en los renglones.

En el proceso de transformación, las pérdidas dependerán del número de términos que se tomen en cuanta para obtener los coeficientes de la transformada, y la mejor manera de evitar pérdidas en un proceso de transformación es el

considerar la mayor cantidad de términos posibles, sin embargo esto conlleva a cálculos más extensivos y por tanto, a mayores tiempos de procesamiento.

2.5.2 Cuantizador

Un cuantizador reduce el número de bits necesarios para almacenar los coeficientes transformados mediante la reducción de la precisión de estos valores, con esto se reduce la redundancia visual. Este es un proceso con pérdidas y es la principal fuente de compresión en un codificador. El cuantizador es una función cuyos valores de salida son discretos y usualmente finitos. Obviamente, este es un proceso de aproximación , y un buen cuantizador es aquel que representa la señal original con la mínima pérdida o distorsión. La cuantización que se realiza en cada coeficiente de manera individual, produciendo la salida correspondiente, se le conoce como Cuantización Escalar. La cuantización, también puede realizarse en un grupo de coeficientes y a esto se le llama Cuantización Vectorial. De igual forma podemos clasificar a un cuantizador según las particiones de las entradas y los niveles de salida. Si el rango de entrada es dividido en niveles igualmente espaciados, entonces se le denomina cuantizador Uniforme, en caso contrario se denomina cuantizador no uniforme. Para un mejor análisis de los diferentes esquemas de cuantización se recomienda consultar [7] .

Un esquema que ha fructificado en modernos algoritmos es el de cuantizar cada sub-banda (obtenida con la transformada *wavelet*) de acuerdo a una tasa de bit asignada, posteriormente codificar los coeficientes que fueron cuantizados mediante el empleo de algún codificador de entropía. En este ramo, algoritmos como el EZW reforzaron la tendencia de los algoritmos de compresión, porque además como resultado se tienen cadenas de datos *embebidas*, y esto es muy útil en la transmisión progresiva.

La mayoría de los métodos para codificación por transformada se basan en las siguientes observaciones:

1) Para la mayoría de las imágenes naturales, la energía se concentra en la sub-banda de baja frecuencia, lo que permite que el resto de las sub-bandas puedan ser cuantizadas e incluso descartadas sin que la imagen se vea afectada severamente.
2) Los coeficientes de cada resolución deberían ser independientes a los coeficientes de otra resolución. Sin embargo existe una correlación entre las sub-bandas de las cuales se puede sacar ventaja, como lo veremos más adelante.

Como podemos ver el algoritmo EZW es en realidad el cuantizador en un sistema de compresión EZW.

2.5.3 Codificador de Entropía

Un codificador comprime sin pérdida los valores cuantizados para alcanzar una compresión mayor. Para esto usa un modelo para determinar de manera precisa las probabilidades de cada valor cuantizado y produce un código basado en dichas probabilidades de manera que la salida resultante sea más pequeña que la entrada. Los codificadores de entropía más usados son la codificación de Huffman y la codificación aritmética, aunque para aplicaciones donde se necesita una rápida ejecución se ha probado que el codificador *run-length* (RLE) es muy eficiente. En este trabajo de investigación se utilizó un código aritmético adaptativo, que se describirá brevemente a continuación.

2.5.3.1 Codificación Aritmética [9] [10]

Este tipo de codificación fue desarrollado por Rissanen [8]. La codificación aritmética se basa en dos principios fundamentales: la probabilidad de un símbolo y el rango de codificación. La probabilidad de los símbolos de la fuente de información determina la eficiencia de la compresión. Además determina los rangos de los intervalos de los símbolos, llamados fuente, para el proceso de codificación. Estos intervalos están contenidos dentro del intervalo normalizado de cero a uno. Los rangos de los intervalos para el proceso de codificación, determinan la compresión a la salida. Básicamente se va dividiendo el intervalo [0,1] sucesivamente hasta obtener un número –dentro de este intervalo- que utilice menos bits que los que hubiesen sido necesarios para representar toda la entrada.

Para entender mejor este tipo de codificación, se muestra un ejemplo:

Supongamos que los símbolos fuente son: {00, 01, 10, 11} y sus probabilidades son: {0.1, 0.4, 0.2, 0.3} respectivamente. Entonces, basándonos en estas probabilidades el intervalo [0, 1) puede ser dividido en cuatro subintervalos;
[0, 0.1), [0.1, 0.5), [0.5, 0.7), [0.7, 1) donde [x,y) denota un intervalo medio abierto, el cual incluye x pero excluye y. Ver tabla2.2 :

Símbolos	00	01	10	11
Probabilidades	0.1	0.4	0.2	0.3
Intervalos de codificación iniciales	[0,0.1)	[0.1,0.5)	[0.5,0.7)	[0.7,1)

Tabla 2.2. Probabilidades, símbolos e intervalos iniciales

Luego para codificar un mensaje de una secuencia binaria:

10 00 11 00 10 11 01

Se toma el primer símbolo 10 del mensaje y se encuentra que su rango de codificación es [0.5, 0.7) . Dado que el rango del segundo símbolo 00 es [0 , 0.1) este es codificado tomando el primer décimo del intervalo [0.5, 0.7) por lo que el nuevo intervalo sería [0.5, 0.52). Similarmente si se codifica el tercer símbolo 11 se obtiene el nuevo intervalo [0.514, 0.52) después de codificar el cuarto símbolo 00, el nuevo intervalo es [0.514,0.5146). Codificando el quinto símbolo 10 se produce el nuevo intervalo [0.5143,0.51442). El siguiente intervalo resultante es [0.514384 ,0.51442) para el sexto símbolo 11. El último intervalo es [0.51439, 0.5143948). La salida comprimida de este mensaje puede ser cualquier número dentro del último intervalo. Este proceso se muestra en la figura 2.2:

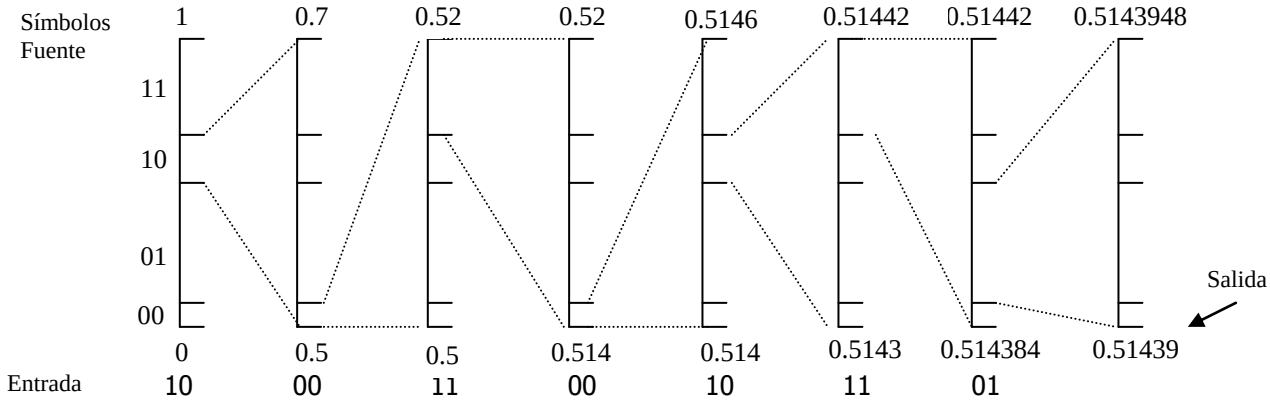


Figura 2.2 .Ejemplo gráfico del proceso de codificación aritmética.
Si se transmite al receptor el número 0.51439 como salida, el decodificador aritmético puede fácilmente descomprimirlo y regenerar el mensaje original, examinando el intervalo al cual el número pertenece.
El proceso de codificación y decodificación se resume en las siguientes tablas:

PASO	Símbolo	Intervalo de codificación	Decisión
1	10	[0.5, 0.7)	El rango del símbolos [0.5,0.7)
2	00	[0.5,0.52)	El primer décimo de [0.5,0.7)
3	11	[0.514,0.52)	Los últimos tres decimos de [0.5,0.52)
4	00	[0.514,0.5146)	El 1er décimo de [0.514,0.52)
5	10	[0.5143,0.51442)	Empezando desde el 5o. Décimo, dos décimos de [0.514,0.5146)
6	11	[0.514384,0.51442)	Los últimos tres décimos de [0.5143,0.51442)
7	01	[0.51439,0.5143948)	Cuatro décimos de

			[0.514384,0.51442) empezando desde el 2o. Décimo
8	Se escoge cualquier número dentro del intervalo [0.51439,0.5143948) como la salida : 0.51439		

Tabla 2.3. Proceso de codificación.

PASO	Rango	Símbolo decodificado	Decisión
1	[0.5, 0.7)	10	0.51439 está en [0.5, 0.7)
2	[0.5,0.52)	00	0.51439 está en el 1er décimo del intervalo [0.5,0.7)
3	[0.514,0.52)	11	0.51439 está en el 7º décimo del intervalo [0.5,0.52)
4	[0.514,0.5146)	00	0.51439 está en el 1er décimo del intervalo [0.514,0.52)
5	[0.5143,0.51442)	10	0.51439 está en el 5o décimo del intervalo [0.514,0.5146)
6	[0.514384,0.51442)	11	0.51439 está en el 7o décimo del intervalo [0.5143,0.51442)
7	[0.51439,0.5143948)	01	0.51439 está en el 4o décimo del intervalo [0.514384,0.51442)
8	El mensaje decodificado es : 10 00 11 00 10 11 01		

Tabla 2.4. Proceso de decodificación.

Para que esto funcione adecuadamente, se asume que tanto el codificador como el decodificador saben la longitud del mensaje para que el decodificador no continúe con el proceso de decodificación. En la realidad se necesita incluir un símbolo especial al término de la cadena , para que, cuando el decodificador reciba este símbolo se termine el proceso.

Algunos problemas asociados con la codificación aritmética son:

1. Dado que no existe ninguna máquina que tenga una precisión infinita, se tendrán problemas de desbordamiento. Muchas de estás máquinas tienen precisión de 16 bit, 32 bits ó 64 bits.
2. El codificador aritmético produce sólo una palabra-código, un número real dentro del intervalo [0,1], para todo el mensaje a ser transmitido. No se puede realizar el proceso de decodificación hasta que se han recibido todos los bits que representan a este número real.
3. La codificación aritmética es sensible a errores, el error en un solo bit puede corromper todo el mensaje.

Estos problemas ha sido analizados por muchos investigadores, y varios esquemas han sido propuestos. Por ejemplo se puede usar un proceso de escalamiento y otro de redondeo para resolver el problema de desbordamiento. La estrategia de escalado vuelve a normalizar cada subintervalo con respecto al rango [0,1) antes de dividirlo según las probabilidades de los símbolos. La estrategia de redondeo garantiza que las pérdidas asociadas a la aritmética de precisión finita no suponen una falta de exactitud de la codificación de los subintervalos.
La codificación aritmética puede ser estática o adaptativa. En la codificación estática las probabilidades de los símbolos fuente son fijas, no así en la otra.
En la codificación adaptativa las probabilidades de los símbolos fuente son estimados de manera dinámica basándose en las frecuencias de cambio de símbolo que se observan en el mensaje a codificar. El proceso para estimar las probabilidades de los símbolos fuente de la parte del mensaje que se ha visto hasta el momento se conoce como modelado. Puesto que usualmente las probabilidades exactas de los símbolos fuente no se conocen o no pueden ser producidas, no podemos esperar que un codificador aritmético logre obtener una eficiencia máxima cuando comprime un mensaje. Lo mejor es estimar las probabilidades durante el proceso. Por lo tanto el modelado dinámico se convierte en la llave para determinar la eficiencia de compresión en un codificador aritmético.
En nuestro trabajo de tesis hemos usado la codificación aritmética adaptativa, por considerarla más adecuada para los fines establecidos.

2.6 Criterios de calidad objetiva

Es importante mencionar que, en cualquier aplicación que involucre compresión de imágenes utilizando un método con pérdidas, debe considerarse que la imagen no se reconstruirá de manera exacta, es decir, después de aplicar un proceso de compresión, se genera un conjunto de datos que solo pueden ser interpretados por la técnica de compresión. Así para poder observar nuevamente la imagen, es necesario aplicar el proceso de descompresión, con en el cual a partir del conjunto de datos obtenido en la etapa precedente, se puede reconstruir la imagen pero ya no de manera exacta.

Es por eso que la evaluación de la calidad entre una imagen original y una imagen reconstruida después de un proceso de codificación-decodificación es seguramente uno de los problemas más cruciales que se pueden encontrar. Se han realizado varias pruebas de carácter subjetivo en las cuales se involucran observadores humanos para evaluar los niveles de calidad en la reconstrucción, sin embargo, estas pruebas suelen ser experimentalmente difíciles y largas, y los resultados dependen de las condiciones de prueba; además, estas pruebas no proporcionan métodos constructivos que permitan mejorar el desempeño de los codificadores y son difíciles de emplear como parte de un diseño. Por esta razón, las medidas

objetivas no sólo alivian las dificultades expuestas, sino ayudan a ampliar el campo de la codificación de imágenes, permitiendo la posibilidad de efectuar ajustes sucesivos para mejorar u optimizar la calidad percibida.

A continuación se enlistan los criterios de calidad objetiva más comunes, los cuales desafortunadamente no se adecuan perfectamente a las propiedades del sistema visual del hombre. Una alternativa se encuentra en ponderar estas medidas objetivas, por medio de algunas funciones de carácter perceptual relacionadas con el modelado del sistema visual del hombre, tal y como lo hace la PQS (*Picture Quality Scale*).

Si I (i , j) e Î (i, j) denotan respectivamente a la imagen original y la reconstruida, entonces se definen los siguientes criterios de calidad objetiva:

El Error Cuadrático Medio (MSE), nos da la información del promedio de error de la imagen. Se calcula de la siguiente manera:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I(i,j)-\hat{I}(i,j)\right)^2$$

donde M y N son los tamaños de la imagen en el sentido horizontal y vertical.

Relación señal a ruido (SNR), se calcula en base al error que se genera entre la imagen reconstruida y la imagen original. Por esta razón este cálculo es independiente del esquema de codificación utilizado:

$$SNR_{dB} = 10\log_{10}\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}I^2(i,j)}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I(i,j)-\hat{I}(i,j)\right)^2}$$

Relación señal a ruido pico (PSNR) está dada por:

$$PSNR_{dB} = 10\log_{10}\frac{255^2 MN}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I(i,j)-\hat{I}(i,j)\right)^2}$$

Sin embargo, la forma más eficiente de evaluar la calidad de una imagen sigue siendo la apreciación subjetiva de dicha imagen. Se sabe que un MSE alto o un PSNR bajo no siempre implica una calidad subjetiva deplorable. Aunque se

establece como regla que una imagen con una PSNR menor a 30 [dB] presenta degradaciones visibles, es decir la calidad visual empieza a decrecer rápidamente.

CAPÍTULO 3

La teoría de las transformadas ha desarrollado un papel clave en el procesamiento de imágenes durante muchos años, y continúa siendo un tema de interés tanto en el trabajo teórico como en el de aplicación en este campo; ya que las transformadas bidimensionales se aplican en la mejora, restauración, codificación y descripción de las imágenes.

En años recientes ha crecido considerablemente el interés con respecto a nuevas técnicas de transformación que, específicamente nos conducen a problemas como: compresión de imágenes, detección de bordes y análisis de texturas entre otros. Es ahí donde encaja la teoría de *wavelets.*

En este capítulo, se revisarán algunas limitaciones de la tradicional Transformada de Fourier y se define una transformada de reciente aparición: la transformada *wavelet* , que promete un mejor desempeño para ciertas aplicaciones. Al principio del capitulo se verán algunos de los desarrollos que han llevado al estado actual del análisis de *wavelet*. Notando las similitudes que tienden a unificar esos diferentes enfoques bajo el título de la transformada *wavelet*. En este capítulo se introducirán los conceptos básicos para la obtención de dicha transformada, de igual manera, se definirá la transformada *wavelet* continua, y la transformada *wavelet* discreta (por simplicidad, se introducirá cada concepto en una dimensión y después se generalizará para dos dimensiones). Por último se ilustrarán algunas de las aplicaciones actuales de la transformada *wavelet* en diversos campos

3.1 Introducción

En el análisis de señales existe un gran número de herramientas que se han ido desarrollando con el paso de los años, entre las que destaca, sin lugar a dudas, la Transformada de Fourier, la que se ha hecho un nombre reconocido gracias a su capacidad de entregar una representación del contenido de frecuencias que posee una determinada señal. Sin embargo, hace no más de 20 años, se han venido desarrollando nuevas herramientas, que permiten realizar un análisis de las señales desde otra perspectiva, surgidas principalmente ante la necesidad de poder analizar señales que no se comportan en forma estacionaria, o que presentan cambios bruscos en intervalos muy pequeños. Estas señales provienen de diferentes áreas de investigación, tales como medicina, sismología, geología, electrónica, desarrollo militar, etc.

Entre estas nuevas herramientas se encuentra la teoría de las funciones llamadas *Wavelet*. Hace no más de 10 ó 15 años que dicha teoría empezó a desarrollarse en el área del procesamiento de señales; los primeros trabajos fueron de Girard en 1982 y luego de Morlet en 1983.

Aunque el primer participante en la carrera de las *wavelet* fue un matemático húngaro llamado Alfred Haar, que introdujo en 1909 las funciones que actualmente se denominan "*wavelets* de Haar". Estas funciones consisten simplemente en un breve pulso positivo seguido de un breve pulso negativo.
En 1930, John Littlewood y Richard Paley, de la Universidad de Cambridge, demostraron que la información local sobre una onda, como la duración de un impulso de energía, se puede recuperar mediante la agrupación de los términos de sus series de Fourier en "octavas".

En 1946 Dennis Gabor realizó una adaptación a la transformada de Fourier para tratar de superar la deficiencia que esta presenta, es decir, cuando se observa una señal producto de la Transformación de Fourier, resulta imposible determinar *cuándo* ocurre un determinado evento, o *cuándo* está presente una determinada frecuencia. Si las propiedades de la señal que se está analizando no cambian demasiado en el tiempo, es decir, si se trata de una *señal estacionaria*, esta desventaja no resulta muy relevante (como en el caso que de señales periódicas, por ejemplo). Sin embargo, un importante número de señales de interés en el mundo físico presentan características no estacionarias, o transitorias, tales como una tendencia a los cambios abruptos, comienzos o finales de eventos, etc. A menudo, estas características no estacionarias resultan ser las secciones más interesantes de las señales. Entonces la nueva adaptación hecha a la transformada de Fourier, consistió en analizar la señal por secciones. El proceso se lleva a cabo mediante la multiplicación de una ventana temporal con la señal. Gabor le llamó a esta técnica *Short-Time Fourier Transform*, (*STFT*) [11]; la cual convierte una señal del dominio temporal, en otra señal bidimensional en tiempo y frecuencia. Desafortunadamente, esta transformación no resuelve del todo las limitaciones que la FT presenta, ya que solamente se puede obtener información de la señal con una precisión limitada, la cual está acotada por el tamaño de la ventana.
Por otro lado para 1976 los físicos de IBM Claude Galand y Daniel Esteban descubren la codificación subbanda, una forma de codificar transmisiones digitales para el teléfono. Posteriormente en 1984 un artículo publicado conjuntamente por Morlet y Grossmann introduce por primera vez el término "*wavelet*" en el lenguaje matemático. Un año después, Yves Meyer, de la Universidad de París, descubre las primeras *wavelets* ortogonales suaves.

Siguiendo con esta evolución, en 1986 Stéphane Mallat en la Universidad de Pennsylvania, demuestra que la base de Haar, las octavas de Littlewood-Paley, las frecuencias de Gabor y los filtros subbanda de Galand y Esteban están todos relacionados con algoritmos basados en *wavelets.*

Ingrid Daubechies construye las primeras *wavelets* ortogonales suaves con una base sólida, en el año de 1987. Sus *wavelets* convierten la teoría en una herramienta práctica que cualquier científico con una formación matemática mínima puede programar y utilizar fácilmente.

3.2 Análisis Multirresolución

Muchos de los desarrollos que precedieron al análisis con *wavelets,* fueron llamados con el término de análisis multirresolución. Estos desarrollos fueron hechos para combatir las limitaciones de la transformada de Fourier y actualmente se hace referencia a ellos como un trabajo que guió al análisis moderno de las *wavelets.*

Por otro lado, la teoría de "bancos de filtros" ofrece un buen medio para representar en multirresolución a las señales formadas por componentes oscilatorias, como lo son las notas musicales. Por regla general, estas componentes incluyen bastantes ciclos de oscilación dentro de su duración.

Sin embargo, en el análisis de imágenes las componentes que fueron encontrados de interés a veces no son verdaderamente oscilatorias, estas incluyen un ciclo o a veces solamente parte de un ciclo. Como por ejemplo las líneas, los puntos y los bordes. También se ha observado que los objetos en una imagen ocurren a diferentes escalas. Un borde, por ejemplo, puede ser una transición repentina de blanco a negro, o bien otra que ocurre gradualmente sobre una distancia considerable. En general, un enfoque multirresolución es una representación de una imagen que busca explotar esta última idea.

La cartografía ilustra este enfoque. Los mapas son comúnmente dibujados a diferentes escalas, la escala de un mapa es la razón del tamaño actual de un territorio sobre la del tamaño de su representación en el mapa. Las escalas grandes corresponden a vistas globales (no detalladas), por ejemplo, se pueden apreciar los continentes y los océanos, mientras que los detalles como las calles de las ciudades están por debajo de la resolución del mapa. A menores escalas los detalles se vuelven visibles y las características más grandes se pierden. Similarmente, en términos de frecuencia, las bajas frecuencias (altas escalas) corresponden a la información global de una señal (es decir, lo que generalmente marca la tendencia de la señal), mientras que las altas frecuencias (bajas escalas) corresponden a información detallada de patrones ocultos de la señal (los que usualmente tienen una duración reducida de tiempo).

Así pues, las transformadas *wavelet* permiten desarrollar este tipo de análisis multirresolución y también el análisis tiempo-frecuencia. Una señal se puede

representar en un espacio de dos dimensiones, pero en este caso el eje vertical es el de la escala en vez de la frecuencia. El escalamiento se logra por medio de la dilatación y la contracción de la *wavelet* madre, para formar un conjunto de funciones de base de *wavelet*. La *wavelet* madre ψ (x) es escalada como ψ (x/*a*) (la cual es expandida si *a*>1 y es contraída si *a*<1) para formar un conjunto de funciones de base. A una escala grande, las funciones de base dilatadas buscan características burdas, mientras que para una *a* pequeña se buscan detalles finos [12] .

3.3 Definiciones [13]

Para entender mejor la teoría de *wavelets* a través del análisis llamado de multirresolución, se requiere definir los siguientes conceptos del álgebra lineal:

Definición. Un espacio vectorial real V es un conjunto de objetos llamados vectores, junto con dos operaciones llamadas suma y multiplicación por un escalar en R, que satisfacen los diez axiomas enumerados a continuación:

i. Si $x \in V$ e $y \in V$, entonces $(x+y) \in V$
(cerradura bajo la suma)

ii. Para toda x, y y z en V, (x+y)+z = x+(y+z)
(ley asociativa de la suma de vectores)

iii. Existe un vector $0 \in V$ tal que para todo $x \in V$, x+0 = 0+x =x
(0 es el idéntico aditivo)

iv. Si $x \in \cdot V$, existe un vector – x tal que x+(-x)=0
(-x se llama el inverso aditivo de x)

v. Si x e y están en V, entonces x+y=y+x
(ley conmutativa de la suma de vectores)

vi. Si $x \in V$ y a es un escalar entonces $ax \in V$
(cerradura bajo la multiplicación de un escalar)

vii. Si x e y están en V y a es un escalar, entonces a(x+y) = ax +ay
(primera ley distributiva)

viii. Si $x \in \cdot V$ y a ,b son escalares, entonces (a+b)x=ax+bx
(segunda ley distributiva)

ix. Si $x \in V$ y a, b son escalares, entonces a(bx) = abx

(ley asociativa de la multiplicación por un escalar)

x. Para cada vector $x \in V$, 1x=x

(el escalar 1 se llama idéntico multiplicativo)

Definición. Un subespacio no vacío H de un subespacio vectorial V es un subespacio de V si cumple con:

i. Si $x \in H$ e $y \cdot \in H$, entonces $x+y \in H$

ii. Si $x \in H$, entonces $ax \in H$ para cada escalar a.

Sea H un subconjunto no vacío de un espacio vectorial V y suponga que H es en sí un espacio vectorial bajo las operaciones de suma y multiplicación por un escalar definidas en V. Entonces se dice que H es un subespacio de V y se denota por

$H \subset V$.

Definición. Sean v_1, v_2, ..., v_n, n vectores en un espacio vectorial V. Entonces se dice que los vectores son **linealmente dependientes** si existen n escalares c_1, c_2, ..., c_n no todos cero tales que:

$c_1 v_1 + c_2 v_2 + ... + c_n v_n = 0$

Si los vectores no son linealmente dependientes se dice que son **linealmente independientes**.

Definición. Un conjunto finito de vectores $\{v_1, v_2, ..., v_n\}$ es una **base** para un espacio vectorial V si :

i. $\{v_1, v_2, ..., v_n\}$ es linealmente independiente

ii. $\{v_1, v_2, ..., v_n\}$ genera a V

Definición. Sean u y, v dos vectores en R^n. Entonces el **producto escalar** de u y v, denotado por <u,v>, ·esta dado por:

$<u,v> = u_1v_1+u_2v_2+ ... +u_nv_n$

donde las v_i, u_i son las componentes de los vectores u y v respectivamente.

Definición. Se dice que dos vectores u y, v son **ortogonales** si $<u,v> = 0$

Dado un espacio vectorial U y un subespacio vectorial V, siempre existe un subespacio vectorial W tal que:

- Todo elemento de U puede ser escrito como una combinación lineal de un elemento en V y un elemento en W, denotado U = V + W.

- La representación anterior es única $\Leftrightarrow$ $V \cap W = \{0\}$, esto es llamado descomposición en suma directa de U y se escribe $U = V \dot{+} W$.

En general W no es único, de hecho dado un espacio vectorial U y un subespacio vectorial V es posible escoger W tal que $V \perp W$, esto es, todo elemento de V es ortogonal a cada elemento de W y se conoce como descomposición ortogonal de U y se escribe $U = V \oplus W$.

3.4 Función de escalamiento

Una función de escalamiento ϕ es esencialmente una función $\phi(x)$ que puede ser escrita como una combinación de $\phi(2x-k)$, la cual es una versión de ϕ trasladada y escalada, matemáticamente se expresa así:

$$\phi(x) = \sum_{k=-\infty}^{\infty} p_k \phi(2x - k). \qquad \text{---------- (1)}$$

Conocida como la *ecuación de escalamiento*. La secuencia $\{p_k\}$ es llamada la secuencia de escalamiento de ϕ.

Considérese

$$\phi_{j,k}(x) := \phi(2^j x - k), \quad j, k \in Z. \qquad \text{---------- (2)}$$

una versión de ϕ trasladada en k y escalada en $1/2^j$. Para una j fija se hace referencia al nivel j.

Sea Vj el espacio vectorial generado mediante traslaciones de ϕ en el nivel j.

Obsérvese que según (1) $V_o \subset V_1$, y más aún se genera una secuencia de subespacios anidados V_j tales que

$$\cdots V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \qquad \text{----------(3)}$$

Proposición. Toda función suave en R se puede representar en términos de $\phi_{j,k}$ para algún j suficientemente grande

$$\bigcup_{j\in Z} V^j = L^2(R) \text{ ----------(4)}$$

Las propiedades (1), (2), (3) y (4) dan lugar a lo que se conoce como *Análisis de Multirresolución MRA.*

3.5 Funciones *Wavelet*

Dada una secuencia anidada de subespacios V_j, existen subespacios W_j que son el complemento ortogonal de V_j en V_{j+1} esto es :

$$V_{j+1} = V_j \oplus W_j, \quad j \in Z$$

donde

$$W_j \perp W_{j'} \text{ si } j \neq j'$$

puesto que los espacios están anidados según (3)

$$V_J = V_j \oplus \bigoplus_{k=0}^{J-j-1} W_{j+k} \text{ for } j < J$$

Proposición. Dada una función de escalamiento ϕ en V_j, existe otra función ψ en W_0 llamada *wavelet* tal que :

$$\{\psi_{j,k} : k \in Z\} \quad \text{genera } W_j,$$

donde :

$$\psi_{j,k}(x) := \psi(2^j x - k), \quad j, k \in Z.$$

puede ser escrito en términos de $\phi(2^1x\text{-}k)$ que forma una base de V_1, entonces

$$\psi(x) = \sum_{k=-\infty}^{\infty} q_k \phi(2x - k)$$

conocida como *ecuación de escalamiento para la wavelet*. { q_k } es llamada la secuencia de escalamiento para la *wavelet.*

Con lo anterior se tiene que:

Una *wavelet* es una función ψ que genera un subespacio vectorial W_j el cual es el complemento ortogonal de espacios vectoriales V_j anidados

$$\cdots V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots$$

estos generan un análisis de multirresolución MRA.

3.6 Tipos de transformada *wavelet*

Así como en la transformada de Fourier, para la transformada *wavelet* existen las mismas tres posibilidades: una transformada *wavelet* continua (CWT), una expansión en series con *wavelets* y una transformada discreta *wavelet*. Aunque la situación es ligeramente más complicada, ya que las funciones de base pueden o no ser ortonormales.
Un conjunto de funciones de base de *wavelets* pueden formar una transformada, aunque éstas funciones no sean ortonormales, es decir, por ejemplo, una expansión en series con *wavelets* puede representar una función limitada en banda, con un número infinito de coeficientes. Si esta secuencia de coeficientes es truncada a una longitud finita, entonces la reconstrucción será sólo una aproximación de la imagen original

3.6.1 Transformada *wavelet* continua

Antes de explicar las características del análisis de señales mediante Transformada *Wavelet*, es necesario señalar que una *Wavelet* es una señal (o forma de onda) de duración limitada cuyo valor medio es cero.
La idea básica de la transfomada *wavelet* es la de representar cualquier función f(t) como un conjunto de funciones de base de dichas *wavelets*. El análisis de señales mediante Transformada *Wavelet* descompone la señal en versiones trasladadas (en tiempo) y escaladas de la *Wavelet* original, más conocida como *Wavelet madre*, que servirá como prototipo para todas las ventanas que se emplean en el proceso. Existe una importante cantidad de *familias* de funciones

Wavelets que han probado ser especialmente útiles; entre ellas destacan: *Haar*, *Daubechies*, *Biortogonal*, *Coiflets*, *Symlets*, *Morlet*, *Sombrero mexicano* y *Meyer*, entre otras. En la figura 1 podemos observar algunas de ellas:

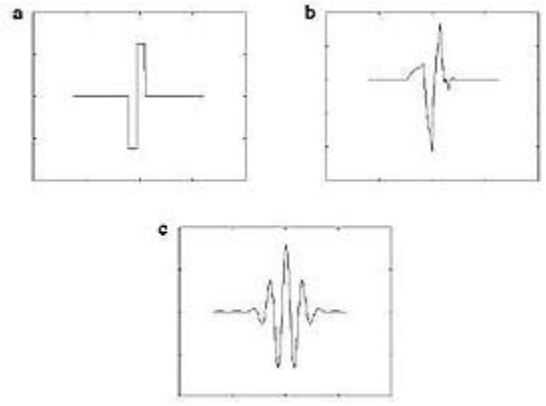


Figura 3.1.(a)Wavelet de Haar, (b) Wavelet de Daubechies, (c) Wavelet de Morlet.

Comparando las *Wavelets* con las funciones sinusoidales (que son la base del análisis de Fourier), mientras que las señales sinusoidales son suaves y predecibles, las *Wavelets* tienden a ser irregulares y asimétricas, por lo que resulta intuitivo pensar que las señales con cambios bruscos serán mejor analizadas mediante *Wavelets* irregulares que a través de suaves sinusoides. Como consecuencia de aquello, es que una de las principales ventajas que provee la Transformada *Wavelet* es su facultad para el análisis de zonas localizadas de señales más extensas.

Así transformada *wavelet* continua se define como:

$$W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(x)\psi(\frac{x-b}{a})dx$$

donde $\psi(x)$ es la *wavelet* madre, dicha función debe verificar ciertas condiciones de admisibilidad, [14] [15], es decir, básicamente se requiere que la función $\psi(x)$ esté bien localizada en tiempo, de media nula y que la transformada $\psi(\omega)$ sea un filtro continuo pasa-banda, con rápido decaimiento hacia el infinito y hacia $\omega = 0$. De hecho la respuesta al impulso de cualquier filtro pasabanda que cumpla con estas características puede servir como una función de base para esta transformada. Cabe notar que el resto de las funciones generadas a partir de esta función por medio de la traslación y el escalamiento, se les denomina simplemente *wavelet.*

Los valores de las transformadas son funciones de dos parámetros **a** y **b**; el parámetro **b** indica el lugar de donde proviene el valor transformado, y el parámetro **a** es el factor de escala que fija el nivel de resolución en el análisis de la imagen. Ambos parámetros son números reales y a>0 . Los *w*(a,b) son también llamados coeficientes de detalles[16], cada uno de estos coeficientes indican los detalles o frecuencias que se pueden hallar en la imagen y los lugares de donde provienen estos detalles, o frecuencias.

3.6.2 Transformada *wavelet* discreta (DWT)

Para aplicar la transformada *Wavelet* a una serie de datos numéricos, se hace necesario implementar una transformada discreta. La idea fue desarrollada por Mallat en 1988 [17], quien diseñó un algoritmo basado en un banco de filtros que permite obtener una transformada *Wavelet* en forma instantánea a partir de los datos de interés. Esto permitió de paso, la unificación con la codificación subbanda.

En la mayoría de las señales son las componentes de baja frecuencia las que le otorgan a la señal la mayor parte de su información, o bien, le dan una especie de identidad a la señal. Mientras que las componentes de alta frecuencia se encargan de incorporar características más particulares. Es por ello que se subdividen las componentes de una señal en dos categorías:

- Aproximaciones (baja frecuencia)
- Detalles (alta frecuencia)

Luego surge la idea de separar estas dos componentes a través de filtros de media banda. Como se muestra en la figura 2:

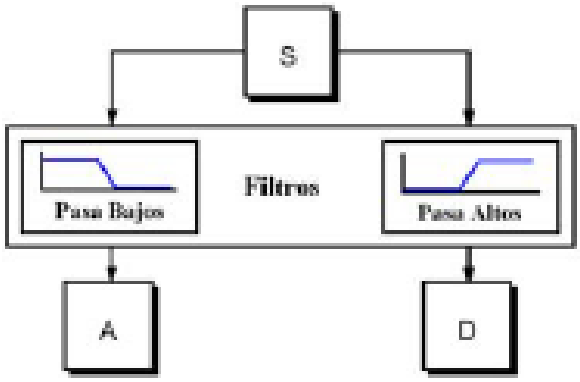


Figura 3.2. Diagrama de descomposición de señales

donde S es la señal que se desea analizar, A la salida del filtro pasobajas y D la salida del filtro pasoaltas. Naturalmente, los filtros son diseñados de tal manera que sean complementarios, es decir, la suma de A y D debe ser S. Si se diseñaran los filtros en forma muy separada en la frecuencia se perdería información, o en caso contrario la banda de entrecruzamiento sería redundante. Sin embargo, este procedimiento tiene la desventaja que se aumenta al doble el numero de datos originales, pues por cada muestra de S se genera un par de muestras (A,D), por lo que el costo matemático y computacional se incrementa. Para remediar esto se propone un método que guarda la mitad de los puntos (A,D), sin perder con ello información de la señal S. Este procedimiento es conocido como submuestreo decimación, el cual en el dominio de la frecuencia equivale a un ensanchamiento del espectro.
El proceso de filtración es realizado a través de la convolución de la señal de entrada con la función de transferencia (discreta) del filtro, lo que puede introducir eventualmente una o dos muestras más.

Sin embargo, para muchas señales de mayor complejidad, no basta con dos bandas de frecuencias (alta y baja), sino que más bien debe hacerse una descomposición de más niveles para poder separar las características y poder analizarlas independientemente. Surge la idea entonces de los filtros multiniveles. Para entender que esto es posible realizarlo en varios niveles, hay que recordar lo que se mencionó respecto a la decimación en el párrafo anterior.

En consecuencia, para esto basta con iterar el proceso de filtrado, es decir, aplicar el mismo procedimiento a las señales de salida de la primera etapa, y así sucesivamente hasta el nivel de precisión que se desee. Lo anterior da origen a una descomposición multinivel conocida como ramificación o árbol de descomposición *Wavelet* . En la figura 3.3 se muestra esta idea:

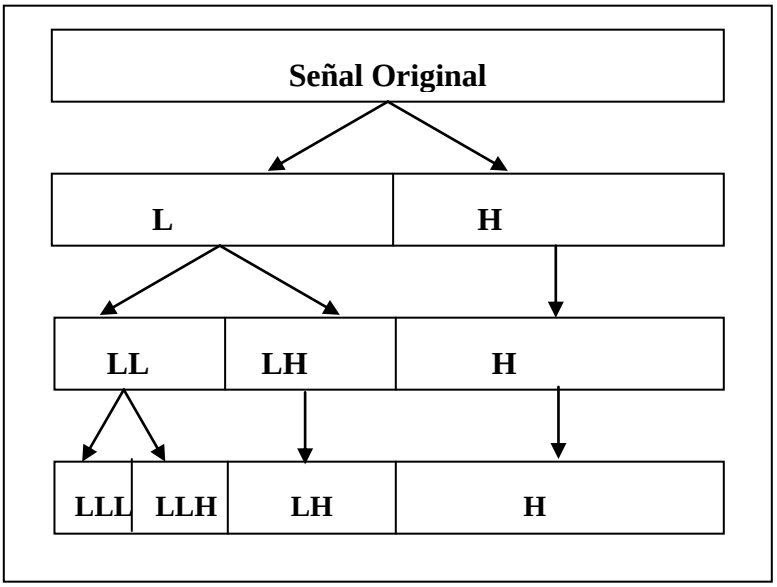


Figura 3.3. Descomposición multinivel para la DWT en una dimensión

El proceso para la Obtención de la Transformada Inversa de *Wavelet* sigue el razonamiento descrito para la Transformada *Wavelet* Discreta, pero en sentido inverso. Esto se ilustra en la siguiente figura:

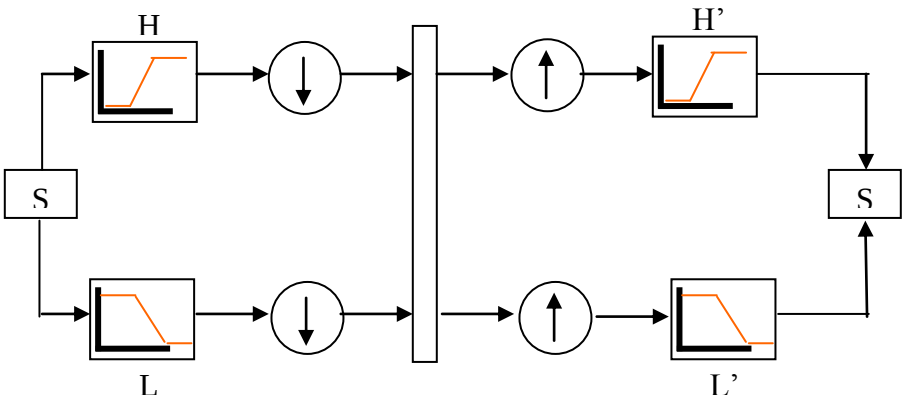


Figura 3.4. Esquema *wavelet* (Descomposición y reconstrucción)

En este caso se debe realizar una sobre-representación de la muestra para compensar el submuestreo realizado en el proceso de descomposición, luego pasa por un proceso de filtrado, para finalmente reconstruir S. La etapa crítica en este proceso es el filtrado, pues la elección de los filtros es determinante en la calidad de la reconstrucción. En [18] se discute el diseño, introduciendo filtros de descomposición H y L (para pasaaltos y pasabajos respectivamente), y sus filtros de reconstrucción correspondientes H' y L', diseñados a partir de una teoría llamada *quadrature mirror filtres (QMF)*, la cual no será analizada en mayor detalle en este trabajo. [19] Es importante mencionar que los filtros utilizados en la DWT inversa no son los mismos que los utilizados en la DWT directa, sino que son filtros diferentes formados a partir de los utilizados en la transformación directa. La relación que existe entre éstos es mostrado con más detalle en [20].

Para transformar imágenes, es necesario una transformada en dos dimensiones. Para realizar esto , es necesario tomar en cuenta la propiedad de separabilidad del núcleo de una transformación lineal [17].Por lo tanto, considerando este principio, en el cálculo de una transformada *wavelet* discreta en dos dimensiones se usa una serie de trasnsformadas *wavelet* discretas de una dimensión:

Paso 1: Se aplica a cada fila de la imagen la DWT en una dimensión.
Paso 2: Se aplica a cada columna de la imagen la DWT en una dimensión.
Paso 3: Se repiten los pasos (1) y (2) en la sub-banda más baja (LL) para crear la siguiente escala.
Paso 4: Se repite el paso (3) hasta alcanzar el número deseado de escalas.

En el diagrama siguiente podemos observar cómo se realiza la DWT:

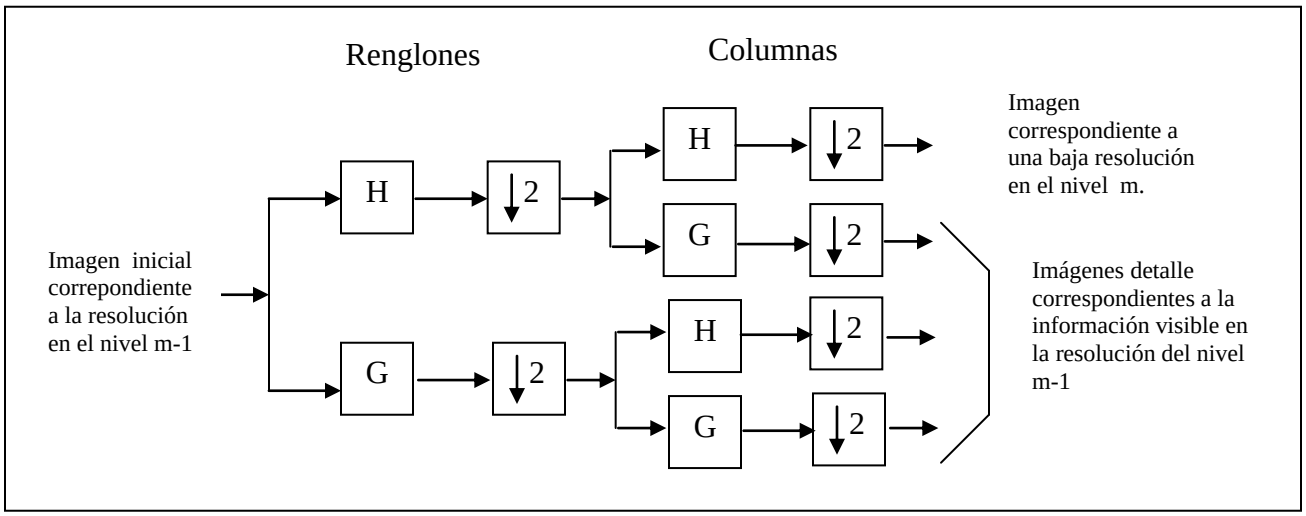


Figura.3.5 Diagrama de la DWT directa para una imagen

En el diagrama se muestra cómo la imagen tiene que ser convolucionada primero por renglones, con las funciones de escalamiento y *wavelet*, H y G respectivamente. Después del bloque de decimación, las salidas tendrán que volver a convolucionarse con los filtros H y G pero ahora por columnas. En el diagrama se indica que la imagen de la entrada, se encuentra en la resolución en el nivel m–1 y que la imagen que resulta de aplicar el proceso de análisis por renglones y por columnas, corresponde a una imagen en la resolución m. Esto quiere decir que si se aplica este proceso una sola vez, la imagen de salida estará en la resolución en el nivel m=1, por tanto la imagen original se encontrará en el nivel m = 0. De hecho la imagen de entrada u original , siempre se encuentra en el nivel m=0 y la última imagen obtenida al ser aplicado el proceso n veces, estará en el nivel m=n. Por ejemplo si el proceso se aplicó tres veces, entonces la imagen, que es producida por la salida de los filtros H, se encuentra en el nivel m=3, por tanto la anterior en el nivel m=2 y así sucesivamente.

Para la resolución, se puede observar que si a una imagen de tamaño N x N, se le aplica el algoritmo de la transformada *wavelet* discreta una sola vez, su imagen de salida de los filtros H será de tamaño N/2 x N/2, es decir, su resolución se disminuyó en un factor de 2, producto del proceso de decimación. De esta forma

las imágenes subsecuentes en la transformada *wavelet,* tendrán cada vez que se realice el proceso recursivamente, la resolución más y más baja.

De igual manera se puede observar en el diagrama anterior que las tres imágenes restantes denominadas imágenes de detalle, son disminuidas en su resolución y nivel *m*. También se nota en el diagrama que la imagen original es convolucionada primeramente por renglones con dos filtros diferentes y después por columnas combinando los dos filtros anteriores. Por ejemplo cuando el proceso se realiza una vez (m=1), se observa que la imagen original es filtrada según cuatro arreglos: paso bajas-paso bajas (HH), paso bajas-paso altas (HG), paso altas-paso bajas (GH) y paso altas-paso altas (GG). A estos arreglos, algunos autores les llaman: aproximaciones pasobajas, detalles horizontales, detalles verticales y detalles diagonales respectivamente, lo cual podemos observar gráficamente en la figura 3.6:

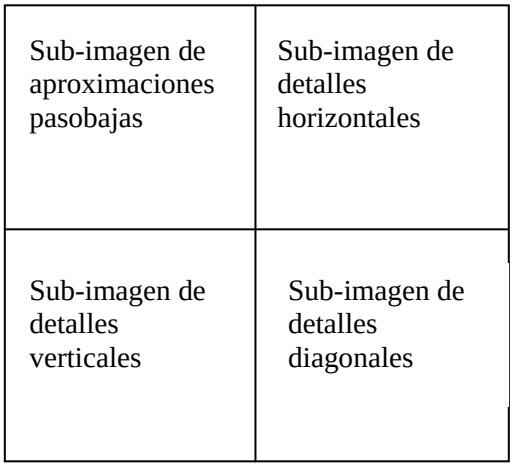


Figura. 3.6 Esquema de una imagen de salida de una DWT, cuando m = 1

En la figura 3.7 se pueden observar los pasos anteriormente descritos. La primera imagen es la imagen original. En la segunda imagen todas las filas han sido transformadas una vez (paso 1). En la tercera y cuarta imágenes se muestra a la imagen transformada con una y dos escalas respectivamente.

La escala *wavelet* se refiere al número de veces que la transformada fue aplicada a la imagen original. En la cuarta imagen es claro que sólo la sub-banda más baja en frecuencias x e y, fue transformada durante la segunda iteración.

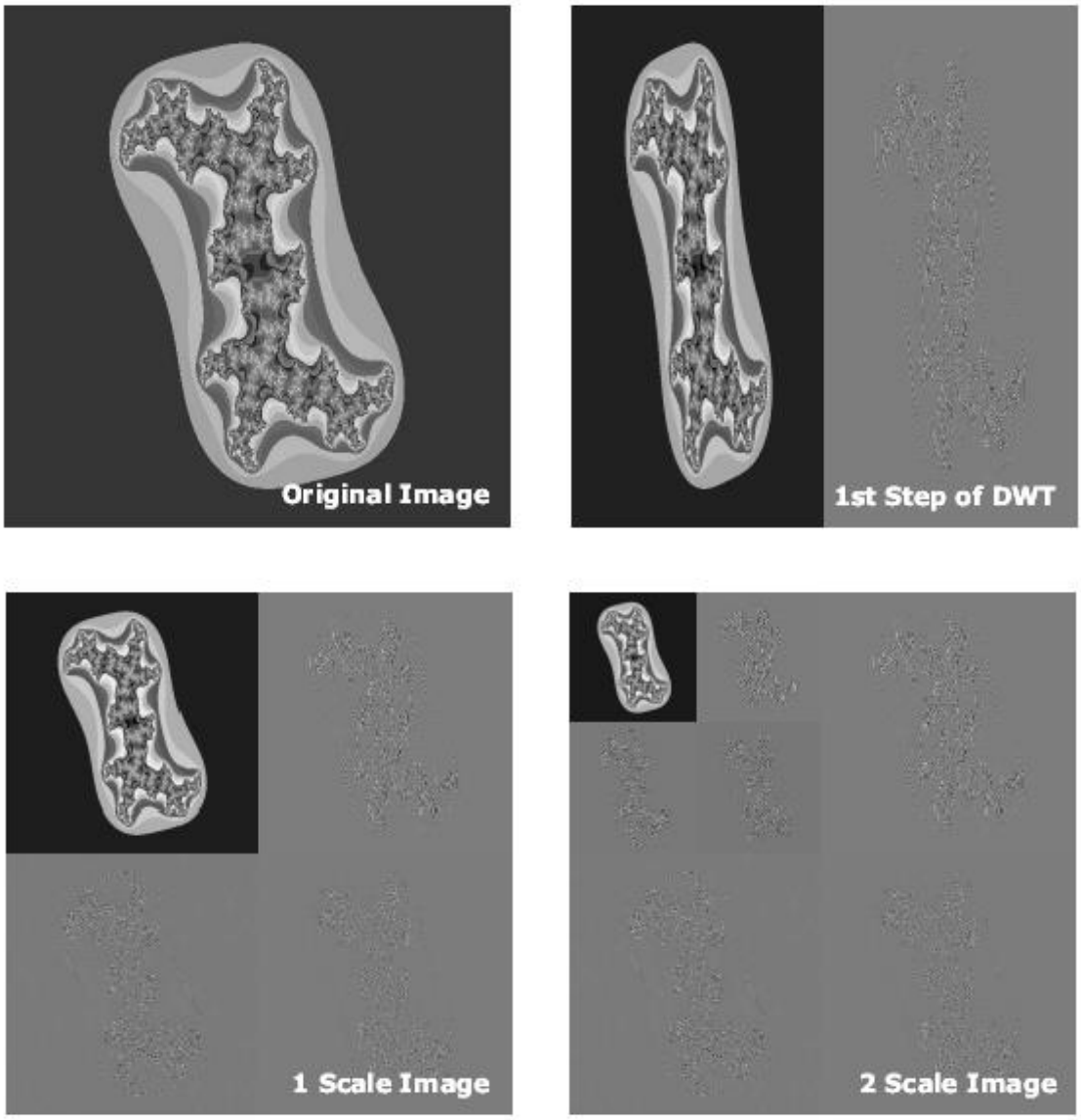


Figura 3.7 DWT en dos dimensiones

En conclusión, la DWT descompone una imagen en un conjunto de pequeñas imágenes ortonormales. Además mientras que el histograma de la escala de grises de una imagen original puede tener cualquier forma, los histogramas de las imágenes transformadas mediante *wavelets* son comúnmente unimodales y simétricos respecto a cero [17]. Esto facilita el análisis de las propiedades estadísticas de la imagen. Por lo que, frecuentemente podemos eliminar aquellos coeficientes que tienen valores pequeños por no poseer mucha energía.

Dado que en este documento de tesis no podemos cubrir toda la teoría de wavelets, se sugiere para una simple y excelente introducción a las *wavelets* ver

[21] [25]. Y para una revisión más profunda de la transformada *wavelet* y los bancos de filtros se aconseja [22] [23].

3.7 Aplicaciones

Como ya se ha mencionado anteriormente, el procesamiento de señales a través de *Wavelets* tiene innumerables aplicaciones en diversos ámbitos de la ciencia e ingeniería. A continuación se presenta una lista de ejemplos:

- *Detección de discontinuidades o de puntos de quiebre en señales (en una o varias dimensiones)*: Resulta de gran utilidad, en especial en el tratamiento de imágenes, en donde interesa detectar la frontera entre colores y formas, o también en sistemas altamente dinámicos en donde interesa determinar *cuando* o *donde* se producen los cambios.
- *Estudio de fractales*: Mediante *Wavelets* se puede reconocer un patrón repetitivo en una señal o imagen, lo que la convierte en una herramienta poderosa en el estudio de fractales.
- *Identificación de frecuencias puras*: Como se trata de una transformada compuesta por una base ortogonal de señales (análoga con la base sinusoidal de Fourier), también pueden ser utilizadas para estudiar el contenido espectral de señales.
- *Eliminación de ruido*: El análisis de señales mediante *Wavelet*, también permite la eliminación o filtrado de ruido tanto en señales unidimensionales como en imágenes (bidimensionales).
- *Compresión de imágenes*: Se trata de una de las aplicaciones más importantes de *Wavelets*, se realiza mediante el análisis en dos dimensiones.
- *Multiplicación rápida de matrices*: La multiplicación de vectores matriciales se realiza en el dominio *Wavelet*. Por ejemplo, si se desea multiplicar una matriz cuadrada de orden *n* en forma sucesiva con *k* vectores *v*, se debe aproximar o transformar los vectores y la matriz por imágenes (en el dominio *Wavelet*), luego realizar la multiplicación, y finalmente aplicar la transformada inversa al resultado obtenido. Si las aproximaciones realizadas son buenas, el error de multiplicación es pequeño con respecto al resultado real por multiplicación ordinaria, pero el tiempo de calculo es notablemente inferior, especialmente en matrices grandes, o en multiplicaciones de múltiples matrices.
- *Aplicaciones en medicina*: Se ha incorporado el análisis con *Wavelets* a señales biológicas, permitiendo interpretar los resultados de exámenes médicos, facilitando el diagnóstico de las enfermedades. Por ejemplo, según [24], se ha aplicado con éxito en el análisis de electroencefalogramas, debido a que en la naturaleza este tipo de señales son altamente no estacionarias (impidiendo el uso de Fourier), las *Wavelets* permiten transformar la señal al dominio tiempo-frecuencia, relacionando el contenido espectral al momento de su ocurrencia. Ello ha sido

aplicado en el diagnóstico de pacientes con Alzheimer, enfermedad que hasta ahora es difícil de diagnosticar. El procedimiento que se realiza es entrenar redes neuronales con los datos obtenidos (a través de *Wavelets* de multirresolución) de pacientes de los que se sabe padecen la enfermedad, para luego introducir las mediciones de aquellos que se realizan exámenes, y de este modo por correlación poder establecer si se siguen los mismos patrones o no.

En general existe una muy diversificada gama de aplicaciones, que crece cada vez más a medida que se incorpora esta tecnología a las distintas ramas de la ciencia.

En este trabajo de tesis, más relacionado con las telecomunicaciones, aplicaremos esta herramienta de análisis con propósitos de compresión

CAPÍTULO 4

Este capítulo está enfocado a describir profundamente el algoritmo implementado en este trabajo de tesis, a saber: *el algoritmo EZW,* el cual se presenta como una alternativa a los métodos que ya se conocen para la compresión de imágenes.
Primeramente se describe al algoritmo así como la estructura de datos de la que saca provecho, la estructura *zerotree*, y que por su propia naturaleza es generada por la transformada *wavelet*.
Posteriormente se detalla el funcionamiento del algoritmo, con todos los pasos que lo conforman. Y para una mejor comprensión, en la sección 4.7 se presenta un ejemplo paso a paso.
Por último, podemos observar algunas simulaciones obtenidas con el sistema de compresión de imágenes desarrollado en esta tesis, y para demostrar que los resultados obtenidos son competitivos se ofrece una comparación con la norma JPEG.

4.1 Descripción del algoritmo EZW *(Embedded Zerotree Wavelet*)

En este trabajo se estudió e implementó un algoritmo de codificación que saca provecho de las estructuras aportadas por la transformada *wavelet* como la correlación que existe entre los coeficientes de cada una de las bandas y tambien de la que existe entre sub-bandas. Este algoritmo es rápido en su ejecución, proporciona buena compresión y una buena calidad cualitativa y cuantitativa en la imagen decodificada, además de que cuenta con una propiedad que se puede explotar para una transmisión progresiva. Esta transmisión progresiva significa que la información codificada es almacenada y transmitida de tal forma que las versiones intermedias de la imagen transmitida contienen aproximaciones completas de la imagen final. Esta es una excelente propiedad para aplicaciones sobre Internet, donde usuarios impacientes pueden observar imágenes intermedias y decidir entre esperar para bajar la imagen completa, o conformarse con una versión incompleta pero rápida.

El método de compresión EZW (embedded *zerotree wavelet*) fue propuesto por J.M. Shapiro en 1993 [26] y trabaja especialmente con la transformada *wavelet* discreta (de ahí la letra W en el acrónimo EZW).
La salida que produce dicho algoritmo es por naturaleza progresiva. Esto significa que mientras más datos se añaden al proceso de compresión, la imagen

reconstruida será más detallada. Un código progresivo es también conocido como código embebido (de ahí la E en EZW).
Este algoritmo utiliza una estructura llamada *zerotree* para codificar los datos (de ahí la Z en EZW) que explicaremos a detalle más adelante.

Una de las características más atractivas de este algoritmo es que la cadena de bits que genera a la salida es completamente escalable lo cual significa que esta cadena de bits puede ser truncada en cualquier punto para alcanzar cualquier tasa de bits seleccionada, y todos los bits que se encuentren arriba de ese punto son completamente útiles para reconstruir una imagen. Esta característica es eficiente en términos del ancho de banda, espacio de almacenamiento, complejidad, tiempo y dinero, según se quiera ver.

El algoritmo EZW se basa principalmente en dos observaciones:

- La primera es que mientras más grande sea el coeficiente obtenido con una transformada *wavelet*, el mismo contiene más información y por lo tanto es más significativo ; es por eso que el algoritmo EZW los codifica primero.
- La segunda observación es que cuando en una imagen se usa la transformada *wavelet* la energía en las sub-bandas disminuye a medida que las escalas disminuyen (una escala baja significa alta resolución), por lo tanto los coeficientes serán, en promedio, más pequeños en las sub-bandas de alta frecuencia que en las sub-bandas de baja frecuencia. Esto demuestra que la codificación progresiva es una elección natural para comprimir imágenes que han sido tratadas con la transformada *wavelet*, ya que las sub-bandas de alta frecuencia sólo añaden detalle a la información promedio de baja frecuencia.

4.2 La estructura del *zerotree*

El *zerotree* es una estructura de datos que permite representar un fenómeno que ocurre en los coeficientes *wavelet*. En 1992 Lewis and Knowles [27] fueron los primeros en introducir una estructura de datos tipo árbol que aprovecha ese fenómeno.
Una transformada *wavelet* transforma una señal del dominio del tiempo al dominio **escala/tiempo**. Esto significa que los coeficientes *wavelet* son bidimensionales. Por lo tanto, si se desea comprimir la señal previamente transformada no sólo codificaremos el valor de dicho coeficiente sino también su posición en el tiempo. Cuando la señal transformada es una imagen entonces su posición en el tiempo se expresa como su posición en el espacio. Después de transformar una imagen la podemos representar usando árboles debido al submuestreo que es posible y que se realiza durante la transformación.

Se dice que un coeficiente *wavelet x* es insignificante con respecto a un umbral *T* dado si $|x|<T$. La estructura *zerotree* se basa en la hipótesis de que si un coeficiente *wavelet* en una sub-banda de baja frecuencia es insignificante con respecto a un umbral, entonces todos los coeficientes de la misma orientación, en la misma posición espacial, en la sub-banda de alta frecuencia, serán probablemente insignificantes con respecto al mismo umbral.

El coeficiente que se encuentra en una escala alta se le denomina *padre*, y todos los coeficientes que corresponden a la misma posición espacial en la siguiente escala más baja, de similar orientación, se les llama hijos. Para un cierto padre, el conjunto de los coeficientes en todas las escalas más bajas con orientación similar y que corresponden a la misma posición son llamados descendientes. De igual manera para un cierto hijo, el conjunto de coeficientes que se encuentran en escalas más altas con orientación similar y que corresponden a la misma posición son llamados ancestros. Esta estructura de datos puede ser visualizada descriptivamente en la siguiente figura:

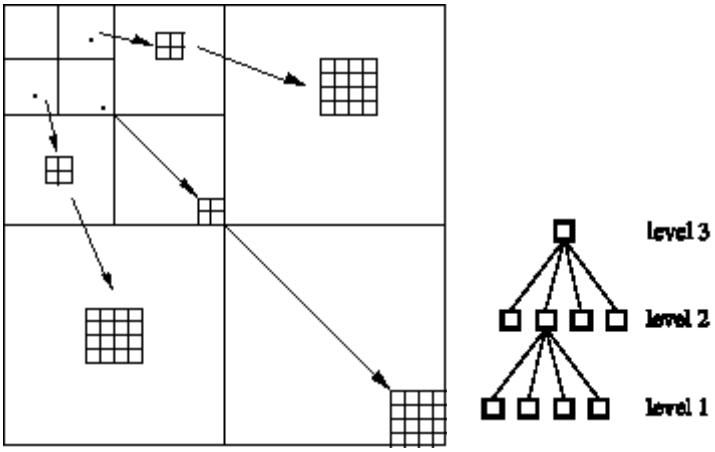


Figura 4.1 Estructura *zerotree*

Como podemos observar en la figura, cada bloque en el nivel más bajo tiene 4 veces más coeficientes que en el nivel anterior. Es decir, cada coeficiente en una escala dada, excepto en la escala más alta, tiene cuatro hijos o descendientes en la correspondiente posición en una escala más abajo. Para la sub-banda de más baja frecuencia, la relación padre-hijo está definida de tal manera que cada nodo padre tiene tres hijos.

Los cuatro hijos inmediatos del coeficiente con posición *(i , j)* están localizados en *(2i-1, 2j-1), (2i , 2j-1), (2i-1 , 2j) y (2i ,2j),* como se muestra a continuación.

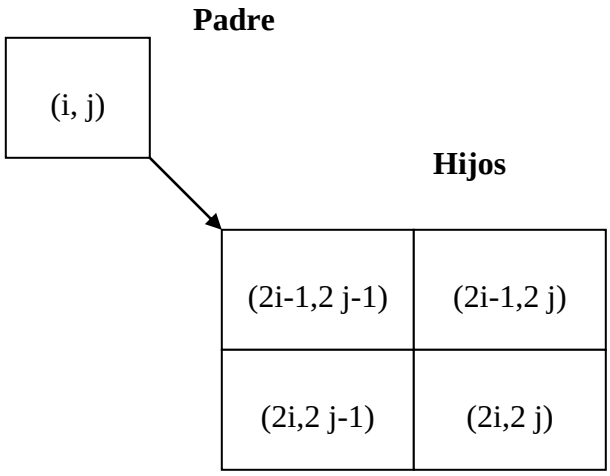


Figura 4.2 Localización de los coeficientes hijos con respecto al padre

Esto es importante para el almacenamiento ordenado de la información de la posición de los coeficientes en el codificador EZW.
En [28] se define a un *zerotree* como un quad-tree en el cual todos sus nodos (hijos) son iguales o menores que la raíz (padre), la cual a su vez es más pequeña que un umbral contra el cual los coeficientes *wavelet* son medidos. Este árbol es codificado con un solo símbolo y es reconstruido por el decodificador como un quad-tree rellenado con ceros.

El barrido de los coeficientes es realizado de tal manera que ningún hijo es barrido antes que su padre.
Para una escala N, el barrido empieza en la sub-banda de más baja frecuencia, denominada LL_N, y sigue en las subbandas HL_N, LH_N y HH_N; terminando el barrido de esta escala se continúa de la misma manera en la escala N-1 como se muestra en la siguiente figura:

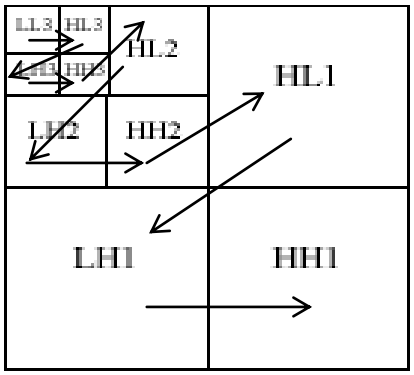


Figura 4.3 Disposición de los coeficientes *wavelet* de tres escalas

Podemos observar que cada coeficiente dentro de una sub-banda dada es barrido antes que cualquier coeficiente en la siguiente sub-banda.
En este trabajo se utilizó el barrido conocido en la literatura como Morton, que se muestra en la figura:

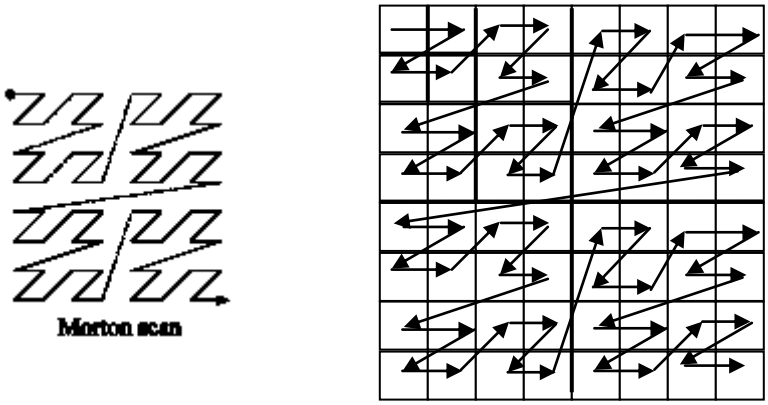


Figura 4.4 Barrido llamado Morton

4.3 Funcionamiento del algoritmo.

El algoritmo EZW comienza calculando la transformada *wavelet* discreta en dos dimensiones de la imagen original.
Para que dicho algoritmo produzca una cadena de bits llamados embebidos, se utiliza una cuantificación por aproximaciones sucesivas (SAQ), junto con la estructura de *zerotree*.
SAQ se relaciona directamente con la codificación de planos de bits de las magnitudes, y su rápida implementación explota esta relación.
Dado un umbral T, se dice que un coeficiente *wavelet x* es insignificante con respecto a T si $|x| < T$.

El umbral inicial T_0 es escogido para que $| x_j | < 2T_0$, para todos los coeficientes transformados x_j.
El SAQ aplica de manera secuencial una secuencia de umbrales T_0 , ..., T_{N-1} para determinar el grado de significancia de los coeficientes. Dichos umbrales son escogidos de tal manera que $T_i = T_{i-1} / 2$. En este trabajo el umbral inicial se calcula con la siguiente ecuación:

$$T_0 = 2^{\lfloor \log (MAX\ |\gamma (x , y)|) \rfloor}$$

Donde :

MAX significa el valor máximo de los coeficientes de la imagen, y γ (x , y) son los coeficientes .

Dado un nivel de umbral T, para determinar si un coeficiente es significativo, es con respecto a este umbral que se comienzan a comparar los coeficientes. La noción de qué es lo que se entiende por un coeficiente "significativo" es importante en la implementación del codificador *zerotree*. Se dice que un coeficiente es significativo con respecto a un umbral dado, si su magnitud excede la del umbral.
Y por otro lado, se dice que un coeficiente *x* es un elemento de un *zerotree* si dicho coeficiente y todos sus descendientes son insignificantes con respecto a T. Un elemento de un *zerotree*, para un umbral T, es una raíz del *zerotree*, si este coeficiente no es un descendiente de una raíz encontrada previamente para el mismo umbral T.
Una raíz del *zerotree* indica que la detección de coeficientes con valores insignificantes en las subbandas de alta frecuencia (baja escala), es completamente predecible.

Un mapa que indica el resultado de una decisión binaria (significativo o insignificante), o ternaria (significativo positivo, significativo negativo o insignificante), se le llama mapa de significancia. Con los umbrales y los coeficientes se forma dicho mapa de significancia y puede ser representado como una cadena de símbolos. Básicamente, los tres elementos usados en esta representación son:

1) Raiz de *zerotree*
2) Cero Aislado
3) Coeficiente Significativo

Para hacer la codificación de la imagen el algoritmo realiza dos pasos: el paso dominante y el paso subordinado . En el paso dominante, la imagen es barrida y se asigna un símbolo para cada coeficiente comparado con el umbral; en el paso subordinado se cuantifican todos los coeficientes significativos que se encontraron en el paso dominante. Estos dos pasos se describirán en detalle más adelante. El

proceso continúa alternando el paso dominante y el paso subordinado. Y el valor del umbral se va dividiendo entre dos antes de cada paso subordinado.

Durante la codificación se tienen dos listas: la lista dominante y la lista subordinada.
La primera contiene las coordenadas de aquellos coeficientes que no se encontraron significativos durante un barrido de la imagen. Este barrido se realiza de tal manera que las subbandas son ordenadas, y dentro de cada subbanda, el conjunto de coeficientes son ordenados también. Este ordenamiento es conocido por el decodificador.

En el codificador, la lista subordinada contiene las magnitudes de aquellos coeficientes que fueron encontrados como significativos. En el decodificador la lista subordinada guarda las coordenadas de los coeficientes significativos, además de la mejor aproximación de cada coeficiente.
Durante un paso dominante, los coeficientes con coordenadas en la lista dominante, son comparados con el umbral T_i, con el objeto de determinar su significancia, y en caso de ser significativo, también se codifica su signo.

Aunque el sistema de codificación embebida descrito es considerablemente más sofisticado que la codificación de planos de bits, la relación con esta codificación nos da una idea de la operación del algoritmo EZW.
Consideremos la cuantización de aproximaciones sucesivas para el caso en que los umbrales son potencias de dos, y todos los coeficientes *wavelets* son enteros. En este caso para cada coeficiente que eventualmente se codifica como significativo, el signo y la posición "de bit" del dígito binario más significativo (MSBD) son medidos y codificados durante el paso dominante. También se considera a los dígitos binarios como una secuencia binaria de decisiones en un árbol binario.
Tomando en consideración a un valor positivo y procediendo de izquierda a derecha, si no se encuentra un "1", se espera que la distribución de la probabilidad del siguiente dígito sea "0". Los dígitos a la izquierda, incluyendo el MSBD, son llamados bits dominantes, y son medidos durante los pasos dominantes. Aquellos dígitos binarios a la derecha del MSBD, son llamados bits subordinados y son medidos y codificados durante los pasos subordinados. Estos bits son menos comprimibles que los bits dominantes.

El diagrama de bloques del algoritmo EZW se presenta a continuación:

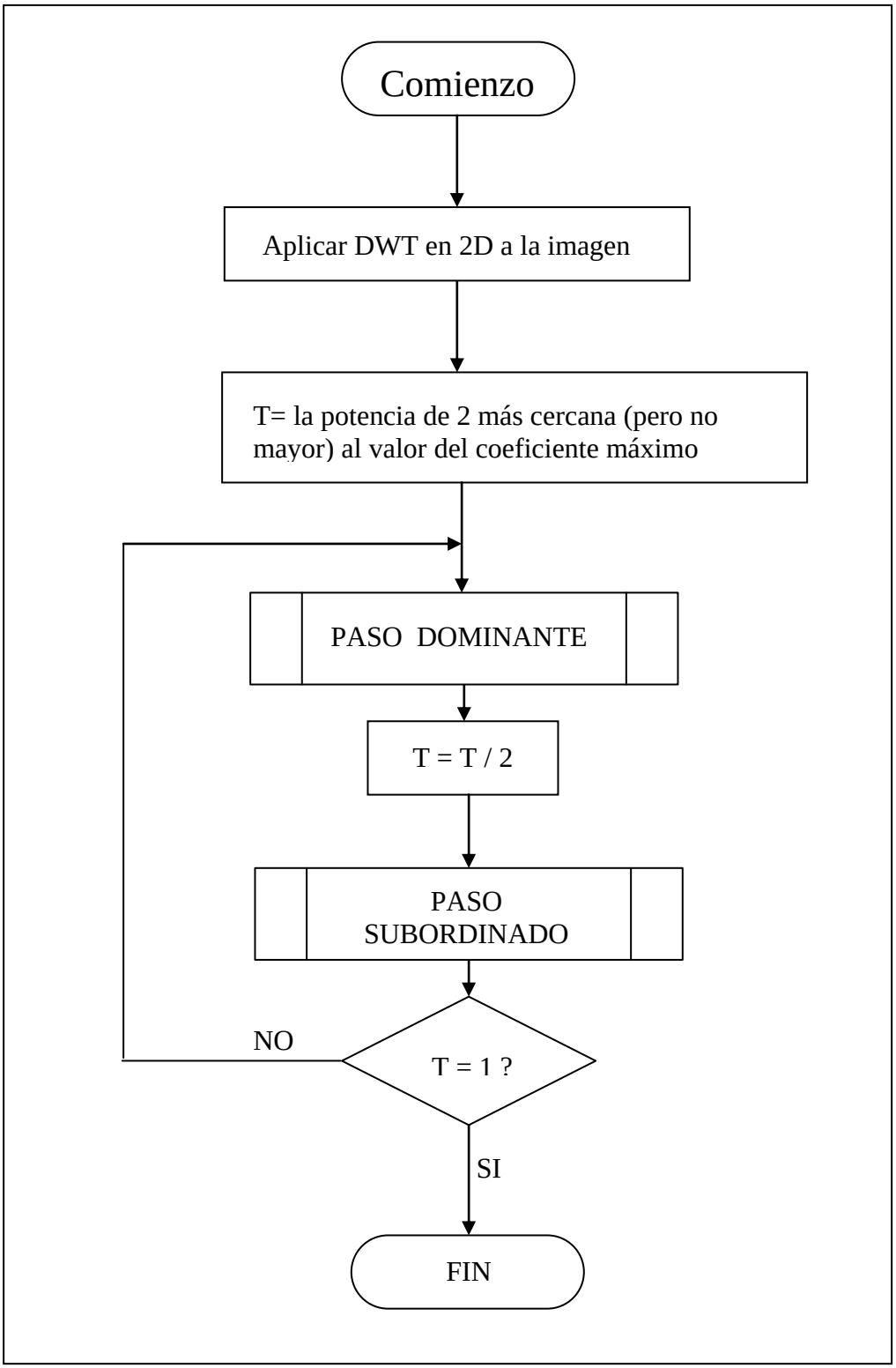


Figura 4.5 Diagrama de bloques del EZW

El algoritmo de codificación *zerotree* produce una secuencia de planos de bits que corresponden a una representación binaria de la imagen previamente transformada. Los bits más significativos son codificados primero y los bits menos significativos son codificados al final.

El proceso se termina cuando el umbral haya alcanzado su valor mínimo, o cuando una función de costo predeterminada haya sido satisfecha. La función de costo más común es aquella que cuenta el número de bits escritos en la cadena de datos comprimida y termina cuando alcanza un valor específico. Ya que el número de bits a la entrada puede ser contado y además los bits a la salida pueden ser controlados; la función de costo puede garantizar tasas de compresión exactas. Otro tipo de función de costo se centra en el error máximo absoluto por píxel, que se tiene entre la imagen original y la imagen reconstruida; y termina cuando este error se encuentra debajo de un margen especificado.

4.4 PASO DOMINANTE

En este paso se barren todos los coeficientes de la imagen transformada y se comparan con respecto a un umbral con el objeto de identificar los coeficientes significativos y los *zerotree*. Este paso produce los siguiente cuatro símbolos:

- P (positivo) cuando un coeficiente positivo resulta ser mayor que el valor del umbral.
- N (negativo) cuando un coeficiente negativo resulta ser más pequeño que el valor del umbral con signo negativo.
- T (raiz) cuando un coeficiente y todos sus descendientes son más pequeños que el umbral.
- Z (cero aislado) cuando un coeficiente es insignificante con respecto al umbral, pero al menos uno de sus coeficientes descendientes es significativo.

Estos símbolos se pasan luego por un codificador de entropía para reducir las redundancias por codificación.

El primer paso a seguir, como ya se dijo, es barrer la imagen previamente transformada y comparar uno a uno sus coeficientes con el umbral inicial. Si el coeficiente es significativo con respecto al umbral, ya sea mayor que el umbral, se extrae de la imagen y se coloca en una lista subordinada; y si el coeficiente es más grande que el umbral se codifica una P; por el contrario, si el coeficiente es más pequeño que el umbral con signo negativo entonces se codifica un símbolo N. Si el coeficiente no es significativo, el próximo paso es revisar si el coeficiente es una

raíz del *zerotree*, en cuyo caso se codifica una T, o si el coeficiente es más pequeño que el umbral pero no es una raíz de un *zerotree*, para este caso se codifica una Z.

Según [29] para mejorar la eficiencia del algoritmo y para prevenir que un coeficiente significativo sea procesado o codificado nuevamente en el siguiente paso dominante, este coeficiente se sustituye por un cero en la imagen.

El siguiente diagrama de bloques muestra el paso dominante

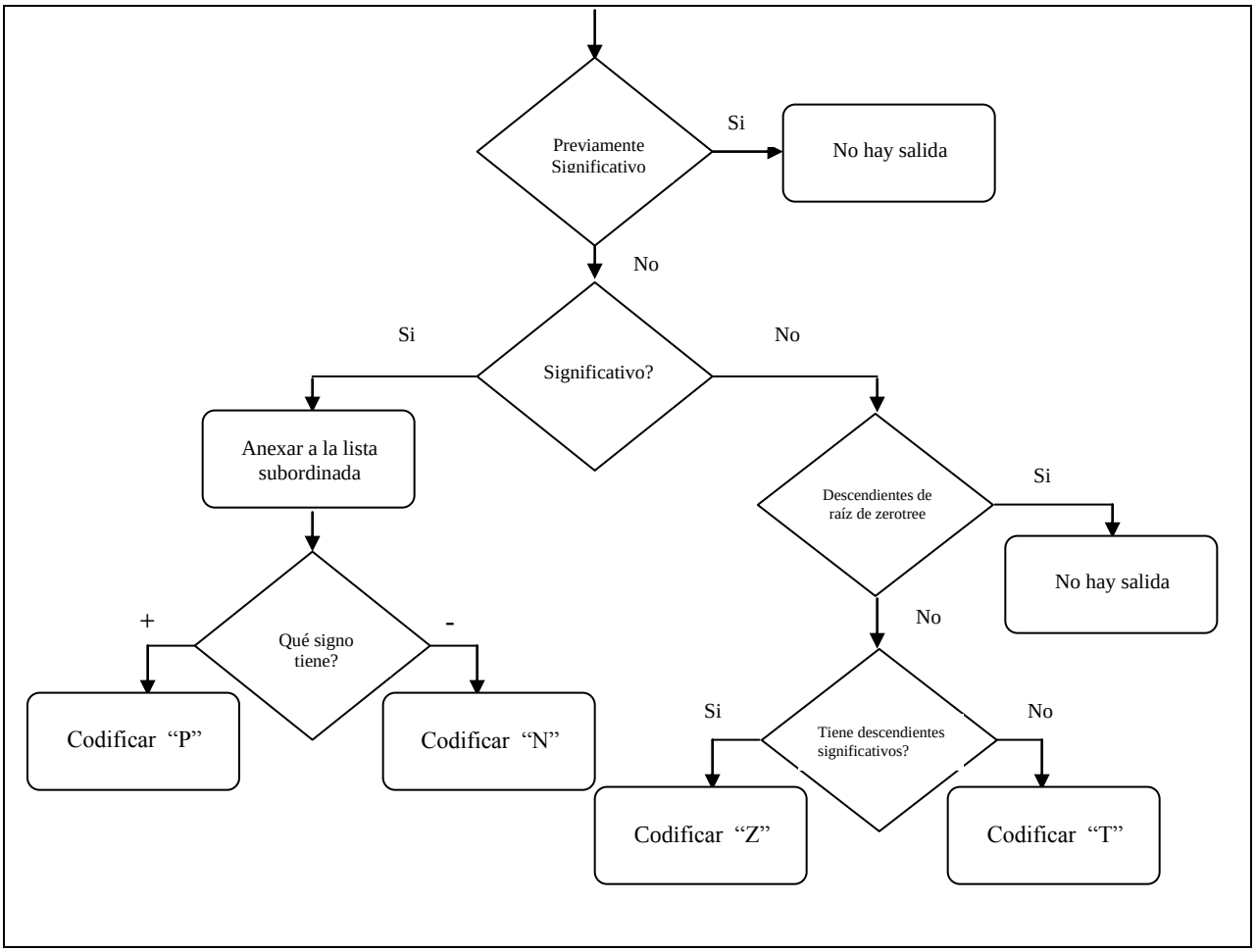


Figura 4.6 Diagrama de bloques del paso dominante

4.5 PASO SUBORDINADO

Después de cada paso dominante, se realiza el paso subordinado sobre la lista subordinada, la cual contiene todos los valores de los coeficientes significativos que fueron encontrados. A este paso también se le llama paso de refinamiento ya que refina el valor de cada uno de los coeficientes significativos. En el paso subordinado se realiza una cuantización de los valores de dichos coeficientes con lo cual se realiza una compresión, ya que con un sólo símbolo se le dice al decodificador el valor del coeficiente en lugar de enviarle el valor exacto de éste.

Para el primer paso subordinado se especifican de entrada sólo dos intervalos en los cuales el coeficiente significativo puede estar situado: en la mitad superior del intervalo, o en la parte inferior de este. Nótese que el ancho de dicho intervalo es exactamente igual al umbral.
La codificación de las magnitudes de cada uno de los coeficientes guardados en la lista subordinada se hace usando un alfabeto binario, con un símbolo "1" se indica que el valor del coeficiente está situado en la mitad superior del intervalo; y con un símbolo "0" se indica que el valor del coeficiente cae dentro del intervalo inferior.

El valor del coeficiente que se encuentra situado en cualquiera de los rangos es cuantizado (desde el punto de vista del decodificador) como el punto medio del intervalo. Al cabo de subsecuentes pasos subordinados, el umbral se va dividiendo a la mitad y por lo tanto para cada paso subordinado el número de intervalos que se tienen son: los intervalos que se tenían en el paso subordinado anterior, los cuales son duplicados y además se tienen dos nuevos intervalos que corresponden al nuevo umbral.

Leyendo el símbolo que el paso subordinado dio a un coeficiente significativo y sabiendo el umbral, el decodificador es capaz de determinar el intervalo en el cual está situado dicho coeficiente y con esto reconstruye el valor del coeficiente. En la operación de decodificación se va refinando y reduciendo el ancho de los intervalos en los cuales el coeficiente pueda ocurrir. El valor de reconstrucción usado puede ser cualquiera dentro del intervalo. Para la mínima distorsión se usa el error cuadrático medio, por lo que se podría usar el centroide de un intervalo usando algún modelo para la función de densidad de probabilidad de los coeficientes. Sin embargo una solución práctica, como ya se había comentado, es la de simplemente usar el centro del intervalo como el valor de reconstrucción.

En la siguiente figura se puede observar con claridad el diagrama de bloques del paso subordinado:

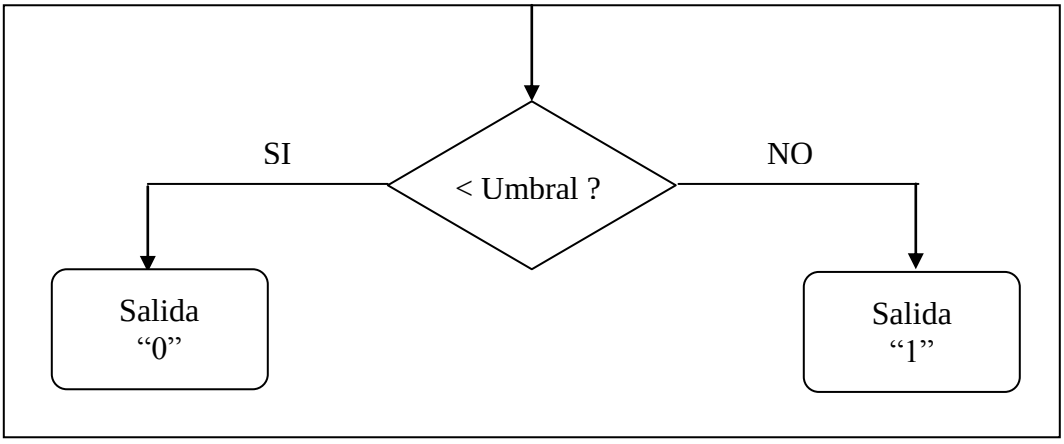


Fig 4.7 Diagrama de bloques del paso subordinado

4.6 Decodificador

La información mínima que debe de tener el encabezado del archivo que llega al decodificador es el número de columnas y de filas que tiene la imagen; el umbral inicial y el número de escalas de la transformada *wavelet*. Este archivo contiene los símbolos producidos por el paso dominante y el flujo de bits producido por el paso subordinado. Con esto el decodificador puede ir construyendo los intervalos y reconstruyendo los valores de los valores significativos. Es importante usar exactamente el mismo orden de barrido usado durante la codificación con el objetivo de preservar la posición de los coeficientes reconstruidos.
El proceso de decodificación, al igual que el de la codificación, puede detenerse cuando se haya alcanzado una tasa de compresión deseada o su equivalente en bpp (bit por pixel).

4.7 Ejemplo del EZW

Para el mejor entendimiento del algoritmo EZW, su funcionamiento será ejemplificado con la ayuda de la siguiente matriz de 8x8, que suponemos es una región de una imagen que ya fue transformada previamente mediante la transformada *wavelet*:

58	41	-44	-17	-8	13	5	8
29	-47	42	-13	3	4	-1	10
22	-14	25	-9	35	-11	6	7
-9	-11	-16	12	-3	6	14	-1
-13	2	0	-11	0	-30	15	7
10	-5	9	4	-7	-1	-9	0
-22	4	-13	3	6	-8	11	5
9	-1	9	-12	2	5	9	-3

Siguiendo los pasos antes mencionados tenemos:

Primer barrido:

Paso 1: Lo primero es determinar el umbral inicial $T_o = 2^{\lfloor \log_2(58) \rfloor} = 32$, este umbral se envía como encabezado del archivo que se enviará al decodificador. De igual manera hay que recordar que no sólo los valores de los coeficientes deben ser codificados, también la posición de los coeficientes debe ser conocida por el decodificador.

Paso 2: Se barren los coeficientes, usando el barrido Morton y se comparan los coeficientes con el umbral inicial (32).

Paso 3: Se comienza a asignar un símbolo para cada coeficiente (Paso dominante), recordando que cuando a un coeficiente se le ha asignado un símbolo T, los coeficientes descendientes no son barridos.
Para el primer paso dominante, se obtiene la siguiente secuencia de símbolos:

D1: PPTNNTPTTTTTZZZZPZZZ

Paso 4: La segunda parte del algoritmo es la cuantización de los coeficientes *P* y *N* obtenidos. Para esto se tiene el siguiente intervalo [32,64). Dividiendo este intervalo en dos partes y asignando el bit "1" al intervalo [48,64) y el bit "0" al intervalo [32,48], se obtiene el siguiente flujo de bits (*58-P, 41-P, -47-N, -44-N, 42-P, 35-P*), donde P y N indican el signo del coeficiente. Por lo que para el primer paso subordinado se obtiene:

S1: 100000

Paso 5: Para finalizar, el codificador envía estos dos segmentos de información al decodificador, y en el decodificador se obtienen los valores de reconstrucción, utilizando los puntos medios de cada intervalo. Por lo que en el decodificador se obtiene:

56	40	-40	0	0	0	0	0
0	-40	40	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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Segundo barrido:

Paso 1: Se obtiene el siguiente umbral $T_1 = T_0 / 2 = 16$.

Paso 2: Se sustituyen con cero aquellos coeficientes que fueron encontrados significativos, para que no vuelvan a ser tomados en cuenta.

Paso 3: Se barren los coeficientes (barrido Morton) y se comparan los coeficientes con el nuevo umbral (16).

Paso 4: Se comienza a asignar un símbolo para cada coeficiente (Paso dominante), de igual manera que en el paso 3 descrito en el primer barrido. Por lo que en este segundo barrido se obtiene la siguiente secuencia:

D2: PNTPTZTPTNTZZZZZZZZZZZZZZZNZZZZNZZZZZZ

Paso 5: Se cuantizan los coeficientes que se encontraron significativos. Como ya se mencionó, el umbral es 16. Para este barrido se tienen tres intervalos de cuantización: [16,32), [32,48) y [48,64), los cuales tienen un rango de 16. Para comenzar con el refinamiento de los coeficientes se subdivide cada uno de estos intervalos, creando dos nuevos intervalos para cada uno de los tres. Por lo que se obtiene el siguiente flujo de bits (*58-P, 41-P, -47-N, -44-N, 42-P, 35-P, 29-P, -17-N, 22-P, 25-P, -16-N, -22-N, -30-N)*. Entonces, en el segundo paso subordinado se obtiene la secuencia:
S2: 1111101001001

Paso 6: El codificador envía estos dos segmentos de información al decodificador, y se obtienen los valores de reconstrucción, utilizando los puntos medios de cada intervalo antes descrito. Por lo que en el decodificador se obtiene:

60	44	-44	-20	0	0	0	0
28	-44	44	0	0	0	0	0
20	0	28	0	36	0	0	0
0	0	20	0	0	0	0	0
0	0	0	0	0	28	0	0
0	0	0	0	0	0	0	0
-20	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tercer Barrido:

Paso 1: Se obtiene el siguiente umbral $T_2 = T_1 / 2 = 8$.

Paso 2: Se sustituyen con cero aquellos coeficientes que fueron encontrados significativos, para que no vuelvan a ser considerados.

Paso 3: Se barren los coeficientes y se comparan los coeficientes con el nuevo umbral (8).

Paso 4: Se comienza a asignar un símbolo para cada coeficiente (Paso dominante). Por lo que en el tercer barrido se obtiene la siguiente secuencia:

D3: NNNNNPNPZZZPZPNZZZZPZNZPZZNPZZPZNZPNZZZPZNZZNZZPZPZ

Paso5: Se comienza con el paso subordinado y con la cuantización de los coeficientes. En este barrido el umbral es 8. En este caso se tienen siete intervalos de cuantización: [8,16), [16,24), [24,32), [32,40), [40,48), [48,56) y [56,64) . Para comenzar con el refinamiento de los coeficientes se subdivide cada uno de estos intervalos, creando dos nuevos intervalos para cada uno de los siete. Entonces, en el segundo paso subordinado se obtiene la secuencia binaria:

S3: 00110010100111100010100011000010110000

Paso 6: El codificador envía estos dos segmentos de información al decodificador, y se obtienen los valores de reconstrucción, utilizando los puntos medios de cada intervalo antes descrito. Por lo que en el decodificador se obtiene:

58	42	-46	-18	-10	14	0	10
30	-46	42	-14	0	0	0	10
22	-14	26	-10	34	-10	0	0
-10	-10	-18	14	0	0	14	0
-14	0	0	-10	0	-30	14	0
10	0	10	0	0	0	-10	0
-22	0	-14	0	0	-10	10	0
10	0	10	-14	0	0	10	0

Por conveniencia, el ejemplo se detendrá en este punto, pero en general, la decisión de la terminación del proceso de codificación depende de la tasa de compresión deseada en la aplicación. Asumiendo que el codificador detiene el proceso de codificación en este punto, entonces el flujo de salida es el siguiente: **Encabezado-D1-S1-D2-S2-D3-S3**. El contenido de cada uno de estos componentes se resume en la siguiente tabla:

8 8 32	Encabezado
PPTNNTPTTTTTZZZZPZZZ / 100000	D1 / S1
PNTPTZTPTNTZZZZZZZZZZZZZZZZNZZZZNZZZZZZ /1111101001001	D2 / S2
NNNNNPNPZZZPZPNZZZZPZNZPZZNPZZPZNZPNZZZPZNZZNZZPZPZ / 00110010100111100010100011000010110000	D3 / S3

Tabla 4.1 Salida del codificador EZW después de los tres barridos.

Es importante tener en cuenta que esta es la salida del segundo bloque de un sistema de codificación por transformación como el mostrado en la Figura 1.1 en el primer capítulo. Ya que en una aplicación real estos símbolos son codificados entrópicamente antes de ser transmitidos, con lo que se alcanza una tasa de compresión más alta.

4.8 Implementación y simulación.

Uno de los principales objetivos de este trabajo de tesis fue la implementación del algoritmo EZW, para probar su desempeño en la compresión de imágenes.
El programa fue escrito en C y se usó el compilador g++ de RedHat 8.
Para la implementación del algoritmo se trató de seguir tal cual, la descripción original en el trabajo descrito en [26]. Sin embargo, se tomó en cuenta la propuesta, que en años posteriores hiciera el mismo autor, para que de una manera más eficiente se pudiera identificar la estructura zerotree durante la codificación con el EZW.

Otras diferencias con el algoritmo original de Shapiro son las siguientes:

1) No hay ordenamiento de los coeficientes en la lista subordinada. En un principio si se implementó, sin embargo no se observó ninguna mejora notable.
2) Para el barrido de los coeficientes se usó, como ya se mencionó, el barrido llamado Morton, para probar el desempeño de esta técnica, ya que en [26] se menciona que Shapiro usó el barrido denominado Raster. Sin embargo en [30] se menciona que la técnica de barrido usada fue el barrido Peano-Hilbert (el único inconveniente de esto es que no hay claridad en cuanto a que técnica de barrido se usó en los resultados presentados en [26]).

Una vez implementado y probado el buen funcionamiento del algoritmo EZW, se prosiguió a completar el sistema de compresión de imágenes. Desde el punto de vista de un diagrama de bloques, antes del bloque del algoritmo EZW viene el de la transformada *wavelet*; para este bloque se uso un programa ya existente. Los filtros utilizados fueron los siguientes:

a) bi9.7
b) dau20
c) dau4
d) dau6
e) haar2
f) haar4
g) haar7
h) haar
i) s10
j) s8

Los cuales corresponden a las familias de *Wavelets* biortogonales (a), Daubechies (b,c,d), Haar (e,f,g y h) y Symlet (i, j).

En cuanto al bloque del codificador entrópico, en la primera versión del programa realizado en esta tesis se implementó un codificador *run-length* (RLE), sin embargo fue sustituido posteriormente por un codificador aritmético adaptativo [31], con el propósito de alcanzar una mayor compresión.

El codificador EZW fue aplicado a diversas imágenes blanco y negro de 8bpp. Una de las imágenes usadas fue Lena de 256 x 256. El desempeño de este codificador sobre la imagen mencionada puede observarse en las siguientes imágenes:

(a)

(b)

(c)

(d)

(e)

(f)

(g)

Figura 4.8 Aplicación del codificador EZW en "Lena". (a) 0.0281982 bits/píxel, PSNR=17.39 (b) 0.0407715 bits/pixel, PSNR=20.12dB, (c) 0.066528 bits/pixel, PSNR=22.14 dB, (d) 0.126282 bits/pixel, PSNR=24.37 dB, (e) 0.679932 bits/pixel, PSNR=30.07dB, (f) 1.276428 bits/pixel, PSNR=33.82 dB, (g) Original Lena 256 x 256, 8 bits/pixel

De igual forma, los resultados obtenidos de la codificación de Lena se resumen en la siguiente tabla:

Umbral	MSE	PSNR(dB)	#bytes	bpp
1024	1186	17.39	231	0.0281982
512	633	20.12	334	0.0407715
256	397	22.14	545	0.066528
128	238	24.37	1032	0.126282
64	128	27.06	2558	0.292908
32	64	30.07	5579	0.679932
16	67	33.82	10508	1.276428
8	11	37.72	18684	2.310425
4	3	43.36	30892	3.733826
2	2	45.12	33188	4.724365

Tabla 4.2 Resultados de la codificación de Lena 256 x 256

En la figura 4.8 podemos observar claramente la naturaleza progresiva que tiene este algoritmo, ya que cada una de las imágenes que forman la figura 4.8 es una versión intermedia de la imagen transmitida y conforme se añaden más datos al proceso de compresión, la imagen reconstruida comienza a detallarse más.

Cabe resaltar que en estas imágenes podemos observar una propiedad interesante: a altas tasas de compresión, como por ejemplo en la imagen (c) de la figura 4.8 aunque la calidad de la imagen no es muy buena (PSNR =22.14 dB), la imagen todavía se puede reconocer. Sin embargo, en el caso de otros esquemas convencionales de compresión de imágenes como JPEG, para estas tasas de

compresión tan altas, la cantidad de bits no es suficiente para codificar los coeficientes y formar una buena imagen.

En lo respectivo al desempeño de los filtros *wavelet* utilizados se observó que ningún filtro se desempeñó regularmemente mejor que los otros en las imágenes procesadas. Aunque en general, en términos cuantitativos, pudimos encontrar que el filtro s10 fue el de mejor desempeño a altas tasas de compresión porque con este filtro se obtuvieron los mejores PSNR, mientras que el filtro haar7 quedó en último lugar, esto se puede explicar por la forma rectangular de la envolvente de las *wavelets* de Haar. En cuanto a los resultados cualitativos podemos observar que con los filtros de Haar aparecen artefactos de bloque debido nuevamente a la forma de onda de la *wavelet*; mientras que con los filtros Symlet, aunque se obtiene una buena evaluación con las métricas objetivas, a bajas tasas de transmisión presentan la desventaja de que pierden gran parte de los detalles de las texturas además de que se tiene también una pérdida de nitidez; sin embargo, el desempeño que presenta permanece uniforme y mejora al disminuir la tasa de compresión.

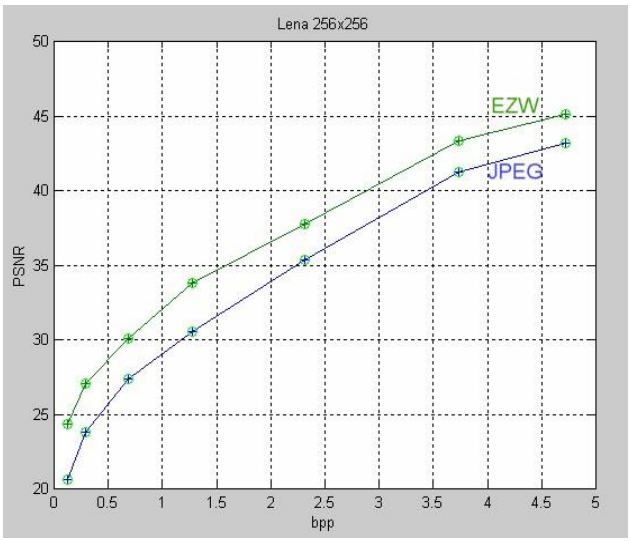
4.9 Comparación con JPEG

En la sección anterior pudimos observar el funcionamiento del algoritmo aplicado a una imagen. Ahora, para demostrar que los resultados obtenidos en este trabajo son competitivos es necesario hacer una comparación con un estándar o norma aceptada internacionalmente.

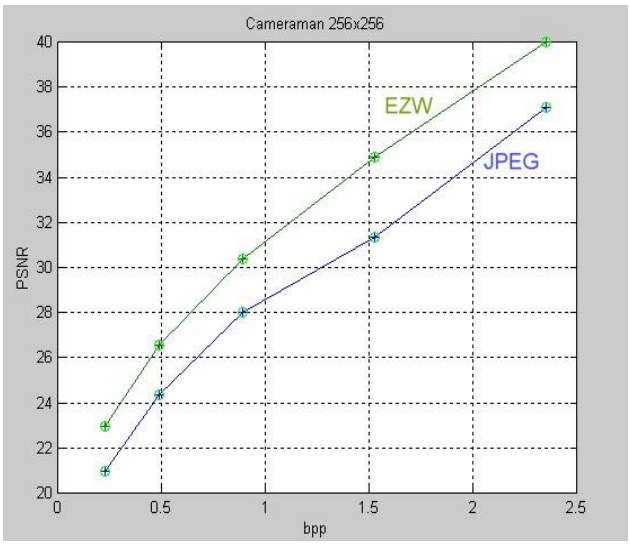
El estándar JPEG (Joint Photographic Experts Group) para compresión de imágenes, utiliza la transformada coseno discreta (DCT) para realizar el análisis de imágenes [32]. Los coeficientes que se generan, se cuantizan por medio de un algoritmo con valores fijos, y se determina de acuerdo a la calidad requerida de la imagen, cuales de estos coeficientes se usarán para la reconstrucción de la misma.

Para hacer dichas comparaciones, en la medición de la calidad de las imágenes comprimidas obtenidas se usaron los criterios de calidad objetiva descritos en el capitulo 1: el error cuadrático medio (MSE) y la relación señal a ruido pico (PSNR).

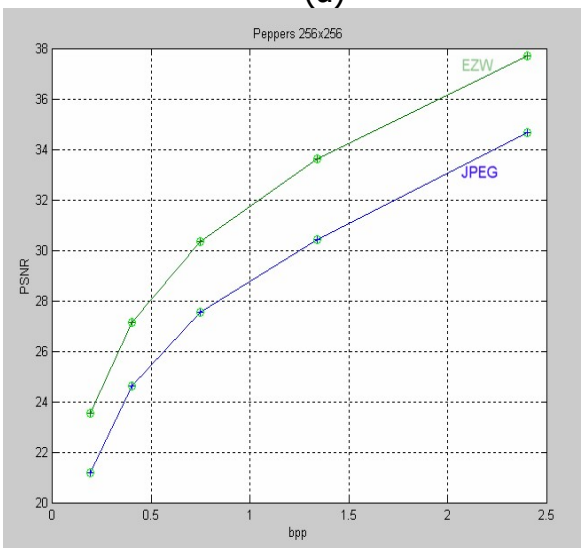
En la figura 4.9 se pueden apreciar las gráficas que se obtuvieron después de realizar la compresión con JPEG y con el sistema de compresión desarrollado en este trabajo para diversas imágenes.


Lena 256x256
PSNR
bpp
EZW
JPEG

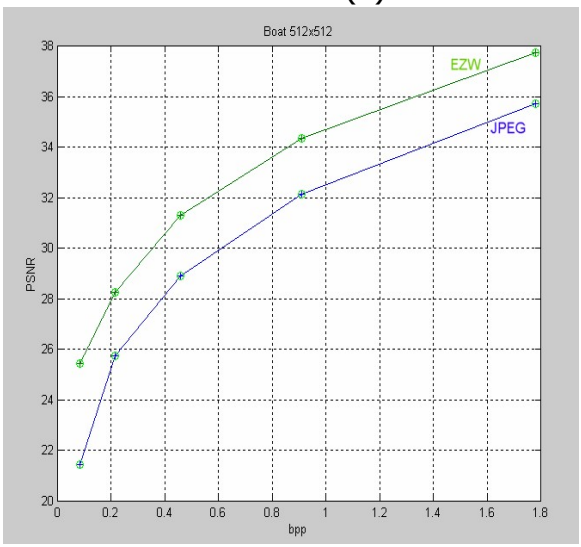
(a)


Cameraman 256x256
PSNR
bpp
EZW
JPEG

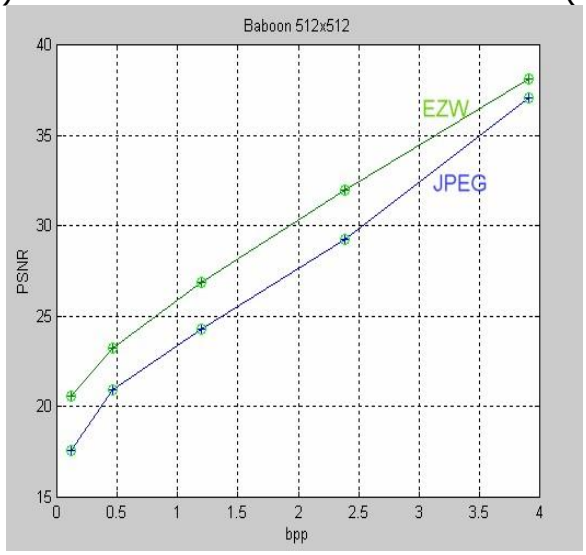
(b)


Peppers 256x256
PSNR
bpp
EZW
JPEG

(c)


Boat 512x512
PSNR
bpp
EZW
JPEG

(d)


Baboon 512x512
PSNR
bpp
EZW
JPEG

(e)

Figura 4.9 Gráficas de comparación del desempeño del EZW contra JPEG (a) Lena 256 x 256, (b) Cameraman 256 x 256, (c) Boat 512 x 512, (d) Peppers 256 x 256, (e) Baboon 512 x 512.

Tomando en cuenta las medidas objetivas, en las gráficas obtenidas se observa que el algoritmo EZW tiene un mejor desempeño en la compresión de imágenes, comparado con la norma JPEG. Por ejemplo se codificó "Lena" usando JPEG, obteniéndose un archivo de tamaño 2230 bytes (0.27bpp) y un PSNR = 23.19 dB; de igual forma se aplicó a la misma imagen el codificador EZW y se encontró que para un archivo del mismo tamaño, 2230 bytes, se obtiene un PSNR = 26.6 dB. Luego se volvió a aplicar el codificador EZW sobre "Lena" para obtener un PSNR de 23.19 dB y se obtuvo un archivo de 969 bytes (0.12bpp). En la figura siguiente podemos observar estas imágenes resultantes para calificarlas cualitativamente:

(a) (b)

(c) (d)

Figura 4.10 Comparación de JPEG y EZW aplicados a "Lena" (a) JPEG 2230 bytes,PSNR= 23.19 dB; (b) EZW 2230 bytes, PSNR = 26.6 dB, (c) EZW 969 bytes, PSNR = 23.19 dB. (d) Imagen original

Definitivamente, se puede observar que comparando las imágenes (a) y (b) , las cuales tienen el mismo número de bytes, la imagen (b) luce visualmente mucho mejor que la (a), debido obviamente a que tiene un PSNR más alto. Para el caso de las imágenes que tienen el mismo PSNR, es más difícil decidir cuál es visiblemente mejor, sin embargo en el caso de la imagen codificada por JPEG son evidentes en toda la imagen, los artefactos de bloque que se producen por la naturaleza propia de la DCT. En cuanto a la imagen codificada con el EZW se observa cierta pérdida de nitidez y existen problemas en algunas regiones como por ejemplo en el borde entre el sombrero y la cara, entre la cara y el hombro, donde se ve difuminado y no se distingue donde empieza uno y donde termina el otro. Es por esto que es muy difícil evaluar la calidad de la imagen mediante apreciación visual, ya que algunas personas probablemente podrían inclinarse a favor de la imagen codificada por JPEG.[33]

En el Apéndice A se muestran los resultados del resto de las imágenes probadas en esta tesis.

Capítulo 5

5.1 Conclusiones y Trabajos Futuros

En los primeros capítulos de este trabajo se hizo incapié en el hecho de que los métodos de codificación y compresión de imágenes están en constante evolución para satisfacer las necesidades de almacenamiento, transmisión, capacidad de canal, etc, que se requieren para las nuevas tecnologías de la información.

En consecuencia, en este trabajo de tesis se desarrolló un codificador basado en uno de los algoritmos más eficientes, el algoritmo EZW (*embedded zerotree wavelet*), con la finalidad de presentar una alternativa a las técnicas conocidas hasta ahora para la compresión de imágenes.

Como pudimos observar en los resultados obtenidos en el capítulo 4, en este trabajo se lograron comprobar algunas propiedades con las que cuenta este algoritmo, entre ellas cabe destacar que el algoritmo no necesita de entrenamiento previo, ni de tablas previamente almacenadas, así como de ningún conocimiento de la imagen a codificar. Otra propiedad, que puede ser explotada para una transmisión progresiva, es la escalabilidad de la cadena de bits que se genera a la salida de dicho algoritmo, lo cual nos permite escoger libremente una tasa de bits y codificar la imagen exactamente con la tasa de bits deseada.

En su conjunto, el codificador desarrollado, que consta de las partes: transformada wavelet – EZW - codificador aritmético adaptativo, proporciona una buena compresión y una buena calidad cualitativa y cuantitativa de la imagen decodificada.

El estudio comparativo realizado que se presenta en el último capítulo, muestra claramente que el codificador diseñado tiene un desempeño muy eficiente, ya que en las gráficas de comportamiento se observa que el algoritmo propuesto es suficientemente competitivo con el estándar JPEG, no importando el tipo de imágenes. Lo cual se pudo comprobar ya que se usaron diferentes imágenes, con características también muy diferentes.

Con estas comparaciones pudimos concluir que el barrido llamado Morton es una buena opción como técnica de barrido. Sin embargo habría que implementar las otras técnicas de barrido existentes, como por ejemplo: Raster, Peano Hilbert, entre otras. Con el propósito de observar si el orden de barrido tiene alguna influencia significativa sobre los resultados finales de la compresión.

Como se ha venido mencionando a lo largo de este trabajo, el algoritmo EZW cuenta con una propiedad que puede ser explotada para una transmisión progresiva, por lo que se propone como proyecto futuro aprovechar el programa desarrollada, escrito en C, e implementarlo en hardware en un procesador de señales, para realizar una transmisión real, por ejemplo: una transmisión de imágenes satelitales; o bien se propone el desarrollo de un formato de compresión de imágenes para internet que aplique tal algoritmo EZW. Para lo cual habrá que estudiar la arquitectura cliente-servidor de la red.

La implementación del esquema de compresión permitió además conocer las bases y diversos usos que tiene esta importante herramienta conocida como la *Transformada Wavelet,* la cual surgió como otra alternativa para ser utilizada en diversas áreas del procesamiento de imágenes. Como se mencionó en el capitulo 4 ninguno de los filtros wavelet usados presentó un rendimiento más sobresaliente en todos los casos probados, de hecho las variaciones en el valor de PSNR entre las imágenes reconstruidas, se puede atribuir a las características inherentes a la imagen original.

A lo largo del desarrollo de este trabajo se comprobó que el desempeño global de la compresión depende de cada una de las etapas que conforman el sistema de compresión de imágenes; por lo que la selección de la *wavelet* apropiada tiene especial importancia, ya que aunque la cuantización sea adecuada (en nuestro caso a través del EZW) y el codificador por entropía sea eficiente, si los filtros tienen un pobre desempeño, el codificador no proporcionará la ganancia suficiente para mantener la calidad de la imagen.

Debido a que el análisis de *wavelets* es demasiado extenso, no se dio especial atención a ese bloque del codificador, y aunque el codificador diseñado arrojó resultados satisfactorios, se recomienda mejorar separadamente este bloque. Esto es posible, dado que la independencia entre los tres bloques de codificación y decodificación, del sistema de compresión desarrollado así lo permite. Entonces es necesario un estudio a profundidad de la selección de *wavelets*, como por ejemplo el uso de sistemas *wavelets* adaptivos [34], cuya idea principal es la de determinar mediante algoritmos, el mejor filtro *wavelet* basado en la naturaleza estadística de la imagen que será codificada. Una vez hecha esta optimización se podrían hacer comparaciones con JPEG 2000 el cual constituye la siguiente versión del formato de JPEG donde se ha reemplazado a la DCT (*discrete cosine transform*) por la compresión basada en *wavelets.*

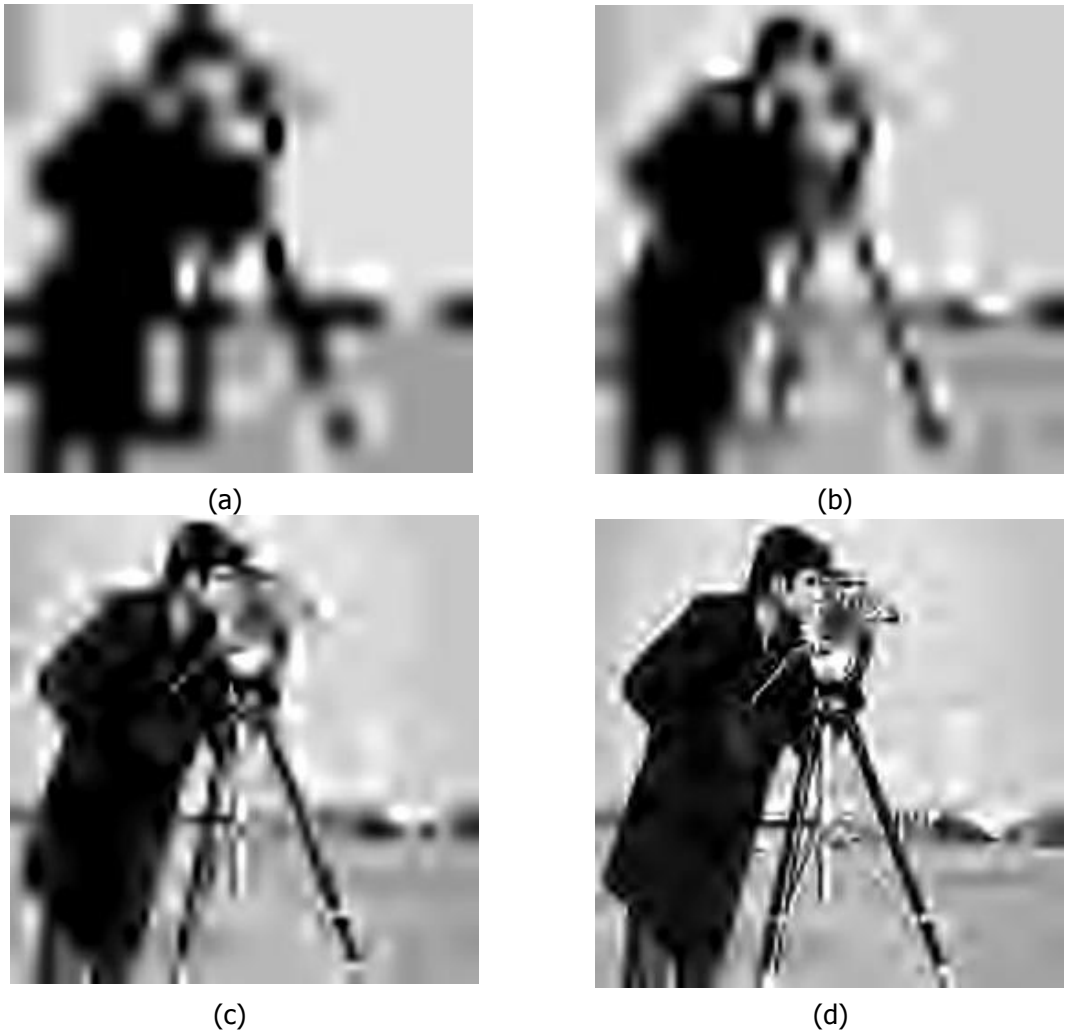
Por otra parte, sabemos que la naturaleza multirresolución de la transformada wavelet la hace un candidato ideal para la transmisión progresiva, sin embargo, cuando el filtrado wavelet es aplicado a una imagen digital (conjunto de pixeles con valores enteros), los coeficientes de los filtros no siempre son números enteros (como es el caso de los filtros utilizados en este trabajo), por lo que se tienen

ciertas pérdidas en la imagen final y como consecuencia se tiene un sistema de compresión de los denominados con pérdidas.

Por último, como se sabe el algoritmo EZW codifica uno por uno los bits de una imagen de coeficientes *wavelet*, y se obtiene una secuencia de planos de bits, por lo que si se utilizan todos estos planos de bits y además en el bloque de la transformación se implementa una de las llamadas *invertible integer-to-integer wavelet transform* [35] (la cual permite la construcción de una versión entera de cada transformada wavelet), sería posible recuperar la imagen original y se tendría así un sistema de compresión sin pérdidas que podría ser aplicado a imágenes médicas, datos sísmicos, imágenes satelitales etc., que no soportan la compresión con pérdidas.

APÉNDICE A: Resultados adicionales

Aplicación del codificador EZW



(a) (b)

(c) (d)

Figura A.1 Aplicación del codificador EZW en "Cameraman". (a) 0.0178222bits/píxel, PSNR=14.09 dB (b) 0.03076172bits/pixel, PSNR=16 dB, (c) 0.04992676 bits/píxel, PSNR=18.01 dB, (d) 0.09655762 bits/píxel, PSNR=20.29 dB.

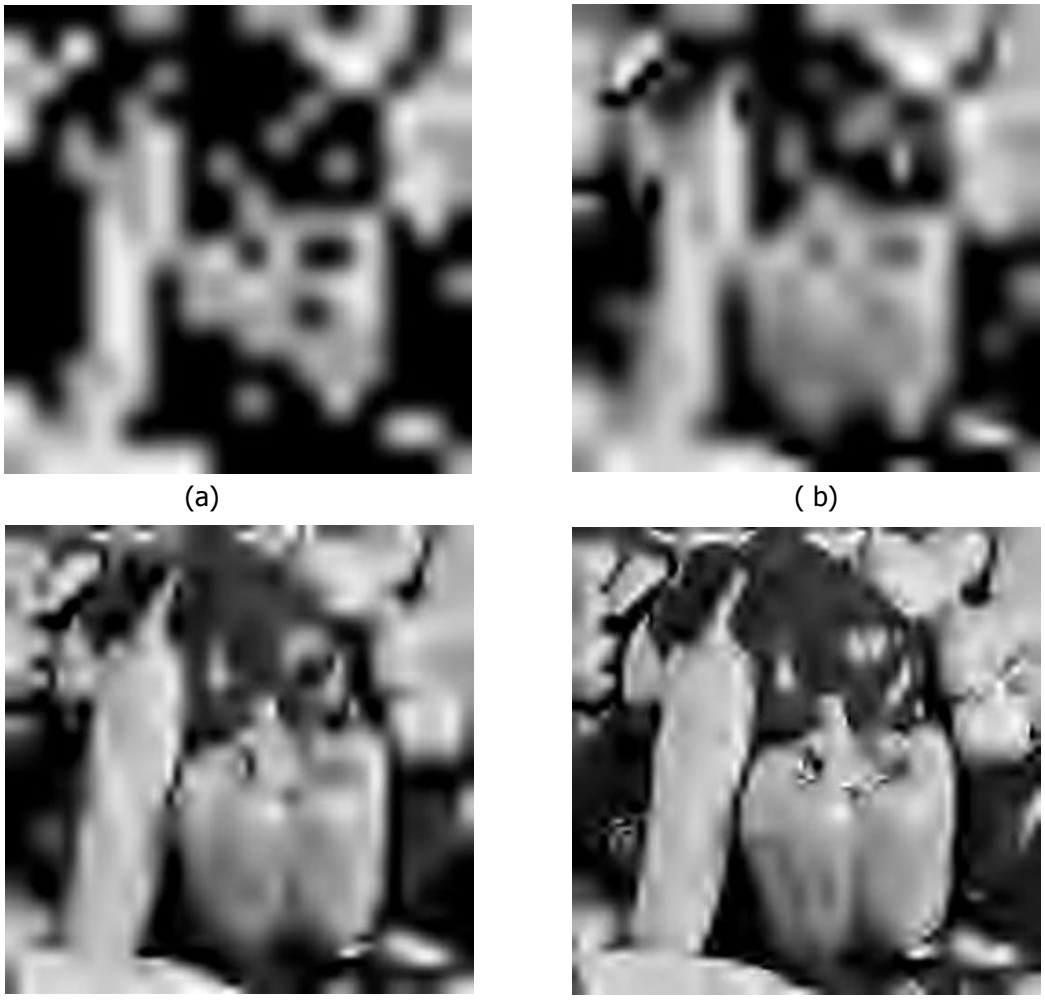
(e)

(f)

(g)

(h)

Figura A.1 (Continuación) Aplicación del codificador EZW en "Cameraman". (e) 0.22943115 bits/píxel, PSNR= 22.96 dB (f) 0.49133301 bits/pixel, PSNR=26.58 dB, (g) 0.8925439 bits/píxel, PSNR= 30.35 dB, (h) Original Cameraman 256 x 256, 8 bits/píxel.



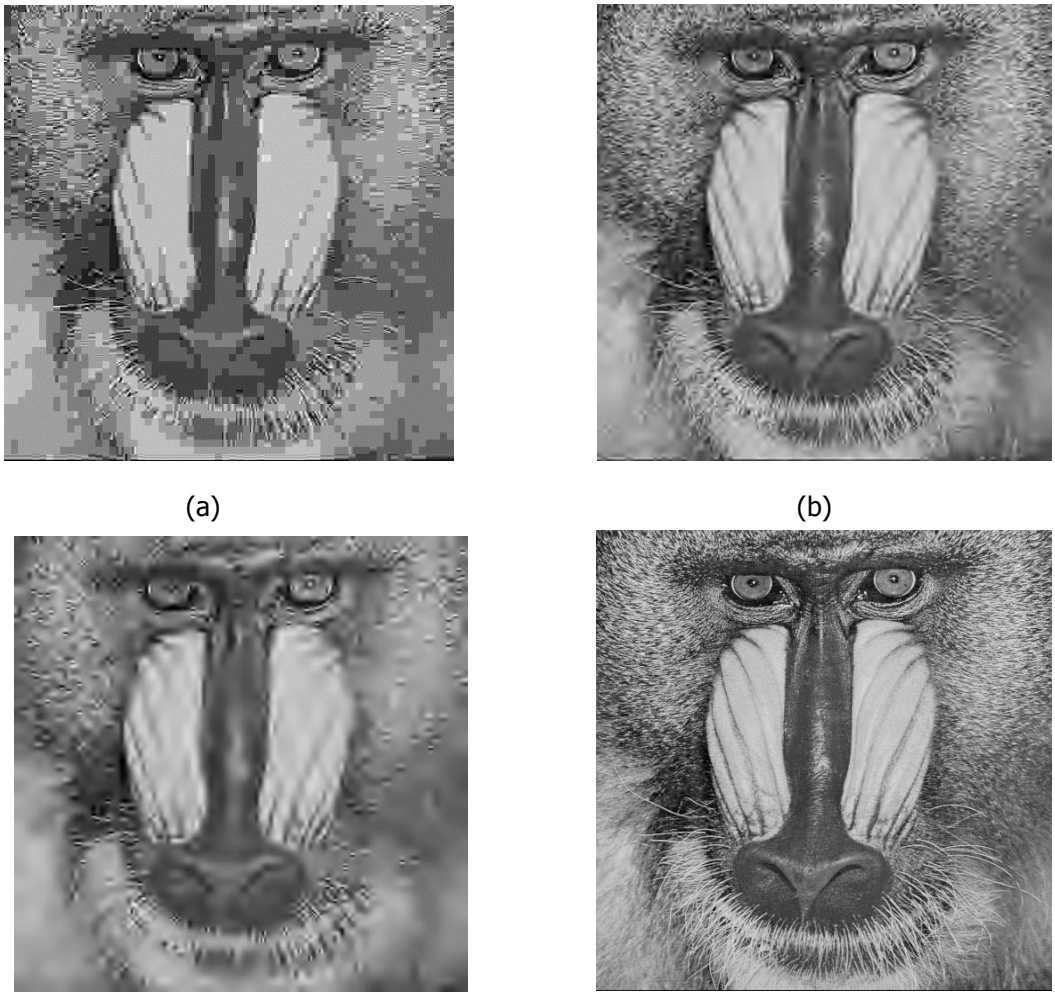
(a) (b)

(c) (d)

Figura A.2 Aplicación del codificador EZW en "Peppers". (a) 0.01855469 bits/píxel, PSNR= 13.09dB (b) 0.03259277 bits/pixel, PSNR=15.8 dB, (c) 0.05090332 bits/píxel, PSNR=18.21 dB, (d) 0.08618164 bits/píxel, PSNR=20.5dB.

Figura A.2 (Continuación) Aplicación del codificador EZW en "Peppers".(e) 0.19219971 bits/píxel, PSNR= 23.55 dB (f) 0.40179443 bits/pixel, PSNR= 27.16 dB, (g) 0.74914551 bits/píxel, PSNR= 30.35dB, (h) Original Peppers 256 x 256, 8 bits/píxel.

Comparación con JPEG



(a) (b)

(c) (d)

Figura A.3 Comparación de JPEG y EZW aplicados a "Baboon" (a) JPEG 3780 bytes, PSNR= 18.25 dB; (b) EZW 3780 bytes, PSNR = 20.61 dB, (c) EZW 1270 bytes, PSNR = 18.25 dB. (d) Imagen original Baboon 512 x 512 .

Figura A.4 Comparación de JPEG y EZW aplicados a "Boat" (a) JPEG 2755 bytes, PSNR= 23 dB; (b) EZW 2755 bytes, PSNR = 25.6 dB, (c) EZW 1341 bytes, PSNR = 23 dB. (d) Imagen original Boat 512 x 512 .

APÉNDICE B: Código

EzwEncoder.c

```
#include "../libs/l_matrix.h"
#include "../libs/l_coordinates.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

void initCoord(int rows, int cols);
void ezwEncoder (Matrix imageMatrix, int lastThreshold_Or_Bits, int typeOfEval);int
initialThreshold (const Matrix *image);
bool dominantPass (Matrix *imageMatrix, int threshold, Matrix *flagMatrix, int
subList[][1048576], int *sizeSubList);
int maxSon (int morton, const Matrix *imageMatrix, int *strMask, int *sizeStrMask, int
threshold);
void subordinatePass(Matrix *compr, const int subList[][1048576], int sizeSubList, int
threshold, int limit);
void evalRanges(Matrix *compr, int limit, int threshold, int value, int morton);int
strBinToValue(const char *bin);
void compFileResults(const char *arch);

FILE *resultsSee;
int totalBits;
Cartesian *coord;
int main( int argc, char *argv[] )
{
  FILE *sizeFile;
  Matrix image;
  totalBits = 0;
  if ( argc == 6 )
  {
        image = readMatrixFromFile(argv[1]);
        printf("image: %s\n%d|%d\n", argv[1], image.row, image.col);
        initCoord(image.row, image.col);
        if ( (resultsSee = fopen(argv[4], "w")) != NULL )
        {
            ezwEncoder(image, atoi(argv[3]), atoi(argv[2]));
            fclose(resultsSee);
            sizeFile = fopen(argv[5], "w");
            fprintf(sizeFile, "Bytes to send: %d\n", totalBits/8);
            fflush(sizeFile);
            fclose(sizeFile);
```

```
        }
        else
            printf("No abrio comp.see\n");
  }
  else
      printf("Usage: EZW_ENCODER inputFile tresholdOn tresholdValue outputFile
sizeFile\n");
  return 0;
}
void initCoord(int rows, int cols)
{
  int a, size = rows*cols;
  coord = (Cartesian *)calloc(size, sizeof(Cartesian));
  for( a = 0; a < size; a++ )
  {
    coord [a] = mortonToCoordinates(a);
  }
}

void ezwEncoder (Matrix image, int lastThreshold_Or_Bits, int typeOfEval)
{
  Matrix flagMatrix;
  Matrix compressed;
  int threshold, sizeSubList = 0, limit;
  int subList[2][1048576];
  bool ok;

  compressed = cleanMatrix();
  compressed.row = image.row;
  compressed.col = image.col;
  threshold = initialThreshold (&image);
  limit = threshold*2;
  flagMatrix = onesMatrix(image.row, image.col);
  printf("%d\n", threshold);
  fprintf(resultsSee, "%d %d %d ", threshold, image.row, image.col);
  while ( threshold >= lastThreshold_Or_Bits )
  {
    ok = dominantPass (&image, threshold, &flagMatrix, subList, &sizeSubList);
    if( ok )
    {
      subordinatePass (&compressed, subList, sizeSubList, threshold, limit);
      threshold /= 2;
    }
    else
      threshold = 0;
  }
```

```
}

int initialThreshold (const Matrix *image)
{
  int aux, max;

  max = maxSubMatrix(image, 0, 0, image->row-1, image->col-1);
  printf("Max: %d\n", max);
  aux = (int) (log10(1.0*max) / log10(2.0));
  return ( (int)pow(2, aux) );
}

bool dominantPass (Matrix *imageMatrix, int threshold, Matrix *flagMatrix, int
subList[][1048576], int *sizeSubList)
{
  Matrix localMask;
  char Z[3] = "00", T[3] = "01", N[3] = "10", P[3] = "11"; //Los valores de los caracteres
en 2 bits
  int strMask[132000], sizeStrMask;
  int limit = imageMatrix->row * imageMatrix->col - 1;
  int i, j = *sizeSubList, coeffValue, maxValue, z;
  bool done = false;

  localMask = onesMatrix( imageMatrix->row, imageMatrix->col );


  for (i = 0; i <= limit; i++)
  {
if(localMask.elem[coord[i].x][coord[i].y]==1&&flagMatrix->
elem[coord[i].x][coord[i].y] == 1 )
    {
      done = true;
      coeffValue = imageMatrix->elem[coord[i].x][coord[i].y];
      if( coeffValue >= threshold )
      {
        fprintf(resultsSee,"P");
        totalBits += 2;
        subList[0][j] = coeffValue;
        subList[1][j] = i;
        j++;
        imageMatrix->elem[coord[i].x][coord[i].y] = 0;
        flagMatrix->elem[coord[i].x][coord[i].y] = 0;
        (*sizeSubList)++;
      }
      else if( coeffValue <= -threshold )
      {
```

```
          fprintf(resultsSee, "N");
          totalBits += 2;
          subList[0][j] = coeffValue;
          subList[1][j] = i;
          j++;
          imageMatrix->elem[coord[i].x][coord[i].y] = 0;
          flagMatrix->elem[coord[i].x][coord[i].y] = 0;
          (*sizeSubList)++;
        }
        else if( coeffValue < threshold )
        {
          maxValue = maxSon(i, imageMatrix, strMask, &sizeStrMask, threshold);
          if( maxValue < threshold && i*4 < limit )
          {
            fprintf(resultsSee, "T");
            totalBits += 2;
            for( z = 0; z < sizeStrMask; z++ )
              setSubMatrixZero(&localMask, coord[strMask[z]].x, coord[strMask[z]].y,
coord[strMask[z]].x-1+2*(z+1), coord[strMask[z]].y-1+2*(z+1));
          }
          else
          {
            fprintf(resultsSee, "Z");
            totalBits += 2;
          }
        }
      }
    }
    return done;
}

int maxSon( int morton, const Matrix *imageMatrix, int *strMask, int *sizeStrMask, int
threshold )
{
    int limit = imageMatrix->col * imageMatrix->row - 1;
    int i = morton, j = 1, auxMax, maxValue = 0;
    strMask[0] = 0;
    *sizeStrMask = j;
    if( i > 0 )
    {
      while( 4*i < limit && maxValue < threshold)
      {
        auxMax = maxSubMatrix(imageMatrix, coord[4*i].x, coord[4*i].y, (coord[4*i].x)-
1+(2*j), (coord[4*i].y)-1+(2*j));
        maxValue = (auxMax > maxValue) ? auxMax : maxValue;
        i *= 4;
```

```
        strMask[j-1] = i;
        j++;
      }
      *sizeStrMask = j-1;
      return maxValue;
    }
    else
      return (maxSubMatrix(imageMatrix, 0, 0, imageMatrix->row-1, imageMatrix->col-
1));
}

void subordinatePass(Matrix *compr, const int subList[][1048576], int sizeSubList, int
threshold, int limit)
{
  int i;
  for (i = 0; i < sizeSubList; i++)
    evalRanges (compr, limit, threshold, subList[0][i], subList[1][i]);

}

void evalRanges(Matrix *compr, int limit, int threshold, int value, int morton)
{
  int i = 1, limInf, sig, auxValue;
  int bitValue = 1;
  int width = threshold / 2;
  limInf = limit – i*width;
  sig = (value < 0) ? -1 : 1;
  auxValue = (value < 0) ? -value : value;
  while ( limInf >= threshold )
  {
    if ( auxValue >= limInf && auxValue < (limit - width*(i-1)) )
    {
      fprintf(resultsSee, "%d", (bitValue > 0) ? 1 : 0 );
      totalBits += 1;
      compr->elem[coord[morton].x][coord[morton].y] = sig*(2*limit - width*(2*i-1)) /
2;
    }
    i++;
    bitValue *= -1;
    limInf = limit - i*width;
  }
}

int strBinToValue(const char *bin)
{
  int i, length, val = 0;
```

```
  length = strlen(bin)-1;
  for( i = length; i >= 0; i-- )
    val += (int)((bin[i]-48)*pow(2,length-i));
  return val;
}
```

ezwDecoder.c

```
#include "../libs/l_matrix.h"
#include "../libs/l_coordinates.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

#define SIZE_SUBLIST 1048576

void initCoord(int rows, int cols);
void ezwDecoder (FILE *bitStr, int lastThreshold_Or_Bits, int typeOfEval);
int dominantPass (FILE *bits, int row, int col, int threshold, Matrix *flagMatrix, int
subList[][SIZE_SUBLIST], int *sizeSubList, Matrix *comp);
void subordinatePass(FILE *bits, Matrix *compr, const int subList[][SIZE_SUBLIST],
int sizeSubList, int threshold, int limit);
void evalRanges(Matrix *compr, int limit, int threshold, int symbol, int morton);

char reconsImage[50];
Cartesian *coord;
int main( int argc, char *argv[] )
{
  FILE *bitStream;

  if ( argc == 5 )
  {
      strcpy(reconsImage, argv[4]);
      if ( (bitStream = fopen(argv[1], "r")) != NULL )
      {
            ezwDecoder(bitStream, atoi(argv[3]), atoi(argv[2]));
            fclose(bitStream);

      }
```

```
      else
            printf("Could not open input file %s\n", argv[4]);
  }
  else
     printf("Usage: decoder inputFileBitStream tresholdOn tresholdValue outputFile\n");
  return 0;
}

void initCoord(int rows, int cols)
{
  int a, size = rows*cols;
  coord = (Cartesian *)calloc(size, sizeof(Cartesian));
  for( a = 0; a < size; a++ )
    coord [a] = mortonToCoordinates(a);
}

void ezwDecoder (FILE *bitStr, int lastThreshold_Or_Bits, int typeOfEval)
{
  Matrix flagMatrix;
  Matrix compressed;
  int threshold, sizeSubList = 0, limit, row, col;
  int subList[2][SIZE_SUBLIST];
  bool ok;

  fscanf(bitStr, "%d %d %d ", &threshold, &row, &col);
  initCoord(row, col);
  compressed = cleanMatrix();
  compressed.row = row;
  compressed.col = col;
  limit = threshold*2;
  flagMatrix = onesMatrix(row, col);
  printf("treshold: %d\nrow: %d\ncol: %d\n", threshold, row, col);
  while ( threshold >= lastThreshold_Or_Bits )
  {
    ok = dominantPass (bitStr, row, col, threshold, &flagMatrix, subList, &sizeSubList,
&compressed);
    if( ok == 1 )
    {
      subordinatePass (bitStr, &compressed, subList, sizeSubList, threshold,
limit);
      threshold /= 2;
    }
    else
      threshold = 0;
  }
  writeMatrixToFile(reconsImage, compressed);
```

```
}

int dominantPass( FILE *bits, int row, int col, int threshold, Matrix *flagMatrix, int
subList[][SIZE_SUBLIST], int *sizeSubList, Matrix *comp)
{
  Matrix localMask;
  char symbol[3];
  int strMask[132000], sizeStrMask;
  int limit = row * col – 1;
  int i = 0, j = *sizeSubList, coeffValue, maxValue, z, t;
  int done = 0;

  localMask = onesMatrix( row, col );

  for (i = 0; i <= limit ; i++)
  {
    if ( localMask.elem[coord[i].x][coord[i].y] == 1 && flagMatrix->
elem[coord[i].x][coord[i].y] == 1 )
    {
      fread(symbol, 1, 1, bits);
      symbol[1] = '\0';
      done = 1;
      if( symbol[0] == 'P' )
      {
        comp->elem[coord[i].x][coord[i].y] = 3*threshold/2;
        subList[1][j] = i;
        j++;
        flagMatrix->elem[coord[i].x][coord[i].y] = 0;
        (*sizeSubList)++;
      }
      else if( symbol[0] == 'N' )
      {
        comp->elem[coord[i].x][coord[i].y] = -3*threshold/2;
        subList[1][j] = i;
        j++;
        flagMatrix->elem[coord[i].x][coord[i].y] = 0;
        (*sizeSubList)++;
      }
      else if( symbol[0] == 'T' )
      {
          z = i;
          t = 1;
          while( 4*z < row*col-1 )
          {
            setSubMatrixZero(&localMask, coord[4*z].x, coord[4*z].y, coord[4*z].x-
1+(2*t), coord[4*z].y-1+(2*t));
```

```
                z *= 4;
                t++;
              }
        }
        else if ( symbol[0] == '0' || symbol[0] == '1' )
        {
              ungetc(symbol[0], bits);
              break;
        }
        localMask.elem[coord[i].x][coord[i].y] = 0;
      }
  }
  return done;
}

void subordinatePass(FILE *bits, Matrix *compr, const int subList[][SIZE_SUBLIST],
int sizeSubList, int threshold, int limit)
{
  int i;
  char symbol[2];
  for (i = 0; i < sizeSubList; i++)
  {
    fread(symbol, 1, 1, bits);
    symbol[1] = '\0';
    evalRanges (compr, limit, threshold, atoi(symbol), subList[1][i]);

  }
}

void evalRanges(Matrix *compr, int limit, int threshold, int symbol, int morton)
{
  int sig, auxValue;
  int width = threshold / 2;
  if ( compr->elem[coord[morton].x][coord[morton].y] >= 0 )
  {
      sig = 1;
      auxValue = compr->elem[coord[morton].x][coord[morton].y];
  }
  else
  {
      sig = -1;
      auxValue = -compr->elem[coord[morton].x][coord[morton].y];
  }
  if ( symbol == 1 )
      compr->elem[coord[morton].x][coord[morton].y] = sig*(auxValue+width/2);
  else if ( symbol == 0 )
```

```
        compr->elem[coord[morton].x][coord[morton].y] = sig*(auxValue-width/2);
}
```

l_coordinates.h

```
typedef struct
{
  int x;
  int y;
} Cartesian;

Cartesian mortonToCoordinates(int morton);
void decToBin(int dec, char *result);
int binToDec(char *bin);
```

l_coordinates.c

```
#include "l_coordinates.h"
#include <string.h>

Cartesian mortonToCoordinates(int morton)
{
  Cartesian result;
  char bin[22] = "00", xBin[11], yBin[11], aux[11];
  int i, length, j = 0;
  decToBin(morton, bin);
  yBin[0] = 0;
  xBin[0] = 0;
  length = strlen( bin );
  if ( (length % 2) != 0 )
  {
    aux[1] = 0;
    aux[0] = '0';
    strcat (aux, bin);
    strcpy (bin, aux);
  }
  for ( i = 0; i < length; i+=2)
  {
    xBin[j] = bin[i];
    yBin[j] = bin[i+1];
    j++;
  }
  xBin[j] = '\0';
  yBin[j] = '\0';
  result.x = binToDec(xBin);
  result.y = binToDec(yBin);
```

```
  return result;
}
/*---------------------------------------------------------------------------*/
void decToBin(int dec, char *result)
{
  int residuo;
  char aux[11];
  if (dec != 0)
    result[0] = 0;
  while ( dec != 0 )
  {
    residuo = dec % 2;
    dec /= 2;
    aux[1] = 0;
    aux[0] = residuo + 48;
    strcat (aux,result);
    strcpy (result,aux);
  }
}
/*---------------------------------------------------------------------------*/
int binToDec(char *bin)
{
  int i, result, LengCad;
  LengCad = strlen(bin);
  result = bin[0] - 48;
  for ( i = 1; i < LengCad; i++ )
    result = result * 2 + (bin[i]-48);
  return result;
}
```

l_matrix.h

```
#define MAX_NUM_COL 1024
#define MAX_NUM_ROW 1024

typedef struct
{
  int row, col;
  int elem[MAX_NUM_ROW][MAX_NUM_COL];
}Matrix;

Matrix cleanMatrix(void);
```

```
Matrix readMatrix(char *message);
Matrix readMatrixFromImageFile(char *fileName);
Matrix readMatrixFromFile(char *fileName);
int writeMatrix(char *message, Matrix A);
int maxSubMatrix(const Matrix *A, int x1, int y1, int x2, int y2);
Matrix onesMatrix(int row, int col);
void setSubMatrixZero(Matrix *A, int x1, int y1, int x2, int y2);
void writeMatrixToImageFile(char *fileName, Matrix A);
void writeMatrixToFile(char *fileName, Matrix A);
```

l_matrix.c

```
#include "l_matrix.h"
#include <stdio.h>
#include <stdlib.h>

Matrix cleanMatrix(void)
{
  int i,j;
  Matrix A;

  A.row = 0;
  A.col = 0;
  for( i = 0; i < MAX_NUM_ROW; i++ )
    for( j = 0; j < MAX_NUM_COL; j++ )
      A.elem[i][j] = 0;
  return(A);
}
/*-------------------------------------------------------------------------*/
Matrix readMatrix(char *message)
{
  int i,j;
  Matrix A;

  A = cleanMatrix();
  printf("%s\n", message);
  printf("Number of rows: ");
  scanf("%d", &A.row);
  printf("Number of columns: ");
  scanf("%d", &A.col);
  for( i = 0; i < A.row; i++ )
    for( j = 0; j < A.col; j++ )
    {
```

```
        printf("Input %d,%d: ", i+1, j+1);
        scanf("%d", &A.elem[i][j]);
      }
  return(A);
}
/*--------------------------------------------------------------------------*/
Matrix readMatrixFromImageFile(char *fileName)
{
  FILE *fin;
  int i,j, aux2;
  unsigned char *RowImage;
  char aux1[10];
  Matrix A;

  A = cleanMatrix();
  if ((fin = fopen(fileName, "rb")) == NULL)
  {
     printf("\n Image File not found \n");
     return(A);
  }
  else
     fscanf(fin, "%s%d%d%d", aux1, &(A.row), &(A.col), &aux2);
  RowImage = (unsigned char *) malloc(A.col);

  for ( i = 0; i < A.row ; i++ )
  {
    fread(RowImage, A.col, 1, fin);
    for ( j = 0; j < A.col; j++ )
       A.elem[i][j] = RowImage[j];
  }
  fclose(fin);
  return(A);
}
/*--------------------------------------------------------------------------*/
Matrix readMatrixFromFile(char *fileName)
{
  FILE *fin;
  int i,j, aux;
  Matrix A;
  A = cleanMatrix();
  if ((fin = fopen(fileName, "r")) == NULL)
  {
    printf("\n Image File not found \n");
    return(A);
  }
  else
```

```
    fscanf(fin, "%d%d", &(A.row), &(A.col));
  for ( i = 0; i < A.row ; i++ )
    for ( j = 0; j < A.col; j++ )
    {
      fscanf(fin, "%d", &aux);
      A.elem[i][j] = aux;
    }
  fclose(fin);
  return(A);
}
/*--------------------------------------------------------------------------*/
int writeMatrix(char *message, Matrix A)
{
  int i, j;

  if( (A.row == 0) && (A.col == 0) )
  {
    printf("ERROR: doesn't have any rows nor columns\n");
    return 0;
  }
  printf("\n%s\n", message);
  for( i = 0; i < A.row; i++ )
  {
    for( j = 0; j < A.col; j++ )
      printf("| %d  |", A.elem[i][j]);
    printf("\n");
  }
  return 0;
}
/*--------------------------------------------------------------------------*/
int maxSubMatrix(const Matrix *A, int x1, int y1, int x2, int y2)
{
  int i, j, max = A->elem[x1][y1];
  for( i = x1; i <= x2; i++ )
    for( j = y1; j <= y2; j++ )
    {
      if ( A->elem[i][j] > max )
        max = A->elem[i][j];
      else if ( -A->elem[i][j] > max )
        max = -A->elem[i][j];
    }
  return max;
}
/*--------------------------------------------------------------------------*/
Matrix onesMatrix(int row, int col)
{
```

```
  int i, j;
  Matrix A;
  A = cleanMatrix();
  A.row = row;
  A.col = col;
  for( i = 0; i < row; i++ )
    for( j = 0; j < col; j++ )
      A.elem[i][j] = 1;
  return A;
}
/*-------------------------------------------------------------------------*/
void setSubMatrixZero(Matrix *A, int x1, int y1, int x2, int y2)
{
  int i, j;
  for( i = x1; i <= x2; i++ )
    for( j = y1; j <= y2; j++ )
      A->elem[i][j] = 0;
}
/*-------------------------------------------------------------------------*/
void writeMatrixToImageFile(char *fileName, Matrix A)
{
  FILE *fin;
  int i, j;
  if ((fin = fopen(fileName, "wb")) == NULL)
    printf("Failed to open file.\n");
  else
  {
    fprintf(fin, "P5\n%d %d\n255\n", A.row, A.col);
    for( i = 0; i < A.row; i++ )
      for( j = 0; j < A.col; j++ )
        fwrite(&A.elem[i][j], 1, 1, fin);
  }
}
/*-------------------------------------------------------------------------*/
void writeMatrixToFile(char *fileName, Matrix A)
{
  FILE *fin;
  int i, j;

  if ((fin = fopen(fileName, "w")) == NULL)
    printf("Failed to open file.\n");
  else
  {
    fprintf(fin, "%d %d ", A.row, A.col);
    for( i = 0; i < A.row; i++ )
      for( j = 0; j < A.col; j++ )
```

```
            fprintf(fin, "%d ", A.elem[i][j]);
    }
}
```

5.2 Referencias

[1] K. Jain Anil. Fundamentals of Digital Image Processing. Prentice Hall, Englewood cliffs, New Jersey, 1989.

[2] Weidong Kou. "Digital Image Compresión: Algorithms and Standars" . Kluwer Academic Publishers. 2a Edición. USA, 1995.

[3] González C Rafael. " Tratamiento Digital de Imágenes". Addison Wesley. EU, 1996

[4] Kraniauskas Peter. Transforms in signals and systems. Adisson Wesley. 1999

[5] Ziemer. Principios de Comunicaciones. Editorial Trillas, 1981

[6] A Chellapa, Rama y Sawchuk A lexander. Digital Image Processing. Prentice Hall, 1996

[7] Gersho, A. and Gray, R. M. *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, 1991.

[8] J. Rissanen and G.G. Langdon. ``Arithmetic coding''. *IBM J. Res. Develop.*, Vol. 23, No. 2, pp. 149--162, March 1979

[9] M. Nelson. *The Data Compression Book*. M&T Books, 1991.

[10] H.W. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[11] D Gabor. *Theory of communication*. Journal of Institution of Electrical Engineers (London) 1946; 93:429-57.

[12] Castleman R. Kenneth. *Digital image processing*" Prentice Hall. USA , 1996

[13] Solar González Eduardo, Speziale de Guzmán Leda. *Apuntes de Álgebra Lineal.* Editorial Limusa, 3ª Edición. México, 1996.

[14] Grossman and Morlet . *Decomposition of hardy functions into square integrable wavelets of constant shape*. SIAM J. Appl. Math. 1984

[15] C. K. Chui. *An Introduction to Wavelets*, Academic Press. San Diego . 1992

[16] Jorge Osmar Lugo. *Trasformada Wavelet, Teoría y Aplicación.* Trabajo de Laboratorio IIFaCENA.1999

[17] S. Mallat. *A theory for multiresolution signal decomposition: the wavelet representation.* IEEE Pattern Anal. and Machine Intell., vol. 11, no. 7, pp. 674–693, 1989.

[18] M. Misiti, Y. Misiti, G. Openheim y J. M. Poggy. *Wavelet Toolbox, User's Guide*. Versión 2. The Math Works, Inc. 2000.

[19] Vaidyanathan, P. P, 1987, "*Quadrature mirror filter banks, M-band extensions and perfect-reconstruction techniques*", IEEE ASSP Magazine, July 1987, pp. 4-20

[20] Marc Antonini, Michael Barlaud, Ingrid Daubechies, and Mathieu Pierre. *Image coding using wavelet transform*. IEEE Transactions on Imege Processing. 1 (2):205-220, April 1992

[21] Chan, Y. T. *Wavelet Basics*, Kluwer Academic Publishers, Norwell, MA, 1995.4

[22] Strang, G. and Nguyen, T. *Wavelets and Filter Banks,* Wellesley-Cambridge Press, Wellesley, MA, 1996, http://www-math.mit.edu/~gs/books/wfb.html

[23] Vertterli, M. and Kovacevic, J. *Wavelets and Subband Coding,* Englewood Cliffs,NJ, Prentice Hall, 1995, http://cm.bell-labs.com/who/jelena/Book/home.html.

[24] R. Polikar. *The Wavelet Tutorial.* Durham Computation Center, Iowa State University, USA. 1995. Documento disponible en: http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html

[25] Graps Amara *"An Introduction to Wavelets" .* IEEE Computational Sciences and Engineering, Volume 2, Number 2, Summer 1995, pp 50-61

[26] SHAPIRO, J. M., "Embedded Image Coding Using Zerotrees of Wavelet Coeffcients." IEEE Transactions on Signal Processing, December 1993, Vol. 41, No. 12, pp. 3445–3462.

[27] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 244–250, Feb. 1992.

[28] Clemens Valens, "EZW Encoding," at *http://perso.wanadoo.fr/polyvalens/clemens/ezw/ezw.html*

[29] SHAPIRO, J. M., "A Fast Technique for Identifying Zerotrees in the EZW Algorithm." Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, May 1996, Vol. 3, pp. 1455–1458.

[30] J. M. Shapiro, "Apparatus and method for compressing information," United States Patent Number 5,412,741, Issued May 2, 1995.

[31] Fred Wheeler http://www.cipr.rpi.edu/~wheeler/ac/

[32]William B. Pennebaker, Joan L. Michaell, Van Nostrand Reinhold, *JPEG, Still Image Data Compression Standard*, ITP Inc. (1993).

[33] MONTUFAR-CHAVEZNAVA, R., GARCIA-UGALDE, F. et.al,*comparation of image coding with matching pursuit and high resolution pursuit techniques*, the International Journal of Image Processing and Communications, Vol. 6, N° 1-2, pp. 57-64, 2000. ISSN 1425-140X,

[34] Saha, S. and Vemuri, R. Adaptive Wavelet Coding of Multimedia Images, *Proc. ACM Multimedia Conference,* Nov. 1999, to appear.

[35] Calderbank, R. C., Daubechies, I., Sweldens, W., and Yeo, B. L. Wavelet Transforms that Map Integers to Integers, *Applied and Computational Harmonic Analysis (ACHA),* vol. 5, no. 3, pp. 332-369, 1998, http://cm.bell-labs.com/who/wim/papers/integer.pdf.