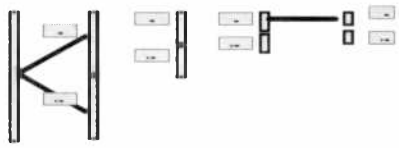
57.

ALGORITMO DE COMPRESIÓN DE IMÁGENES FIJAS UTILIZANDO LA TRANSFORMADA WAVELET



Autores :

Natalia Fournier **Gabriela Castro**

Directores :

Oscar Bria **Claudia Russo**

TRABAJO DE GRADO

Octubre de 1997

AGRADECIMIENTOS

Las autoras agradecen profundamente a sus directores, Oscar Bría y Claudia Russo, por haber proporcionado ayuda constante para el desarrollo del Trabajo.

Asimismo, agradecen y dedican este Trabajo de Grado a sus padres, esposos e hijos, en retribución por todo el apoyo que les brindaron durante el tiempo que demandó el desarrollo del presente estudio.

CONTENIDO

CAPÍTULO I

INTRODUCCIÓN

1- INTRODUCCIÓN

En este trabajo presentamos la aplicación de un algoritmo de compresión de imágenes fijas utilizando la Transformada Wavelet.

La transformada Wavelet es una herramienta conveniente para el análisis multirresolución de señales y en particular se ajusta naturalmente a la compresión de imágenes al adaptar el ancho de banda requerido en forma automática.

Este algoritmo estudia las características de las imágenes en tonos de gris para permitir explotar aspectos importantes del sistema visual humano. El ojo humano es menos sensitivo a las frecuencias espaciales altas (bordes de una imagen) que a las frecuencias espaciales bajas (texturas de una imagen). El método utilizado consiste en codificar con pocos bits los coeficientes que representan frecuencias altas y con más bits los coeficientes de frecuencias bajas.

A continuación describiremos la motivación que nos llevó a su desarrollo.

2- MOTIVACIÓN

La cantidad de información almacenada, transmitida y manipulada por las computadoras ha ido creciendo exponencialmente durante las últimas décadas.

Dos desarrollos recientes han contribuido particularmente a este efecto. Uno es la aparición de sistemas multimedia junto con numerosas aplicaciones que las mismas motivaron. La época en que las computadoras manipulaban sólo números y texto hace tiempo ha terminado y ha sido reemplazada por una era de sonido, imágenes, películas y realidad virtual. Otro desarrollo es la creciente disponibilidad de la Internet, que ha hecho que la información esté disponible para una gran cantidad de usuarios.

Este desarrollo solo fue posible por la rápida evolución del hardware. La performance de la CPUs, discos y canales de transmisión ha crecido exponencialmente. Sin

embargo uno puede fácilmente encontrar ejemplos donde el hardware corriente es inadecuado (técnica o económicamente).

Las técnicas de compresión proveen una solución. Si podemos representar la información en un formato comprimido podemos obviamente:

- ahorrar almacenamiento,
- ahorrar tiempo de CPU,
- ahorrar tiempo de transmisión.

3- OBJETIVOS

Implementar un algoritmo en lenguaje C para la compresión de imágenes fijas utilizando la Transformada Wavelet.

Implementar una interfaz versátil para experimentar con el algoritmo anterior haciendo uso de diversos tipos de filtros, factores de compresión y otros parámetros.

4- ESQUEMA GENERAL DE COMPRESIÓN

4.1- Clasificación de Algoritmos de Compresión

Los algoritmos de compresión de señales intentan minimizar el espacio requerido minimizando de alguna manera la redundancia de información. Las técnicas de compresión de imágenes se dividen en dos grandes grupos:

- con pérdida,
- sin pérdida.

La compresión sin pérdida se aplica cuando la señal reconstruida debe ser exacta a la señal original, por ejemplo archivos de texto. Con este tipo el radio de compresión es limitado.

En el caso de la compresión con pérdida, se permite un error mientras la calidad después de la compresión sea aceptable. Este esquema tiene la ventaja de que se puede lograr radios de compresión mucho más altos que con compresión sin pérdida.

En cualquier esquema de compresión se pretende eliminar la correlación presente en los datos. Existen varios tipos de correlación:

1- Correlación espacial: Se puede predecir el valor de un pixel en una imagen mirando a los pixels vecinos.

2- Correlación espectral: La transformada de Fourier de una señal es a menudo suave. Esto significa que uno puede predecir un componente frecuencial mirando las frecuencias vecinas.

3- Correlación temporal: En video digital, la mayoría de los pixels de dos frames consecutivos cambian muy poco en la dirección del tiempo (por ejemplo: background)

Uno de los estándares para compresión con pérdida es a través de codificación por transformada. La idea es representar los datos usando una base matemática diferente que revele o no la correlación. En esta nueva base la mayoría de los coeficientes serían tan pequeños que podrán ser seteados a cero.

4.2- Sistema Completo de un Compresor de Imágenes

Un compresor de imágenes comprende de una fase de Compresión y otra de Descompresión.

Compresión : Dada una imagen original I, se aplica una transformación T que consiste en mapear los pixels desde un dominio, por ejemplo espacial, hacia otro, por ejemplo frecuencial, en busca de redundancia. Luego en base a un criterio se determina cuáles y cuántos coeficientes deben seleccionarse. Posteriormente, este conjunto pasa por un módulo de cuantificación. En éste proceso se llevan los coeficientes, en base a una tabla específica, a un espacio con menor precisión, requiriendo así menos bits para su representación. Este conjunto de coeficientes cuantificados pasa a un proceso de Codificación ; generalmente se usa un código entrópico que se encarga de asignar de manera eficiente pocos bits a los coeficientes más frecuentes y muchos bits a los menos frecuentes. Finalmente obtenemos un conjunto de datos que representan la imagen comprimida.

Descompresión : Dada una imagen comprimida I', se aplica en forma inversa los procesos inversos de la etapa de Codificación, es decir : Decodificación, Decuantificación y Antitransformación para reconstruir la imagen similar a la original.

5- SISTEMA VISUAL HUMANO

La compresión con pérdida explota aspectos importantes del sistema visual humano ; por ejemplo el ojo percibe mucho más detalle en la luminosidad (brillo) que en la crominancia (color) de la imagen. Por lo tanto en sistemas de transmisión de TV, la luminosidad se muestrea a altas resoluciones (720x480) mientras que la crominancia se muestrea a bajas resoluciones (360x240). Es de esperar que en la imagen comprimida se asignen más bits a la representación de la luminosidad.

Por otro lado en ojo humano es menos sensitivo a las frecuencias espaciales altas, bordes de una imagen, que a las frecuencias bajas, texturas de una imagen ; por ejemplo si en un monitor desplegamos un patrón continuo de pixels blancos y negros alternados uno y otro, el ojo humano tiende a detectar este patrón como un gris uniforme y continuo en lugar del 'mosaico' que tiene a la vista. Esta deficiencia es explotada codificando con pocos bits los coeficientes que representan frecuencias altas y con muchos bits los de frecuencias bajas.

CAPÍTULO II

FUNDAMENTOS MATEMÁTICOS

1- INTRODUCCIÓN

En este capítulo presentamos las bases teórico-matemáticas que respaldan el algoritmo aquí desarrollado. Nos remitimos al concepto original que motiva el estudio de las wavelets, comenzando por la codificación sub-banda.

2- CODIFICACIÓN SUB-BANDA

Existen diferentes esquemas de compresión, en muchos de los cuales la eficiencia depende de las características de los datos. Desafortunadamente, la mayor parte de las salidas de la fuente exhibe una combinación de características, lo cual hace difícil la elección de un esquema de compresión adecuado a ellas.

En la codificación sub-banda se descompone la señal en sus partes constituyentes; después se puede usar la técnica de codificación más adecuada a cada constituyente para mejorar la performance de la compresión. Más aún, cada componente de la salida de la fuente puede tener diferentes características perceptibles. Por ejemplo el error de cuantificación que es perceptualmente objetable en una componente podría ser aceptable en una componente distinta de la salida de la fuente; por lo tanto un cuantificador más grosero que usa menos bits puede ser usado para codificar la componente que es perceptualmente menos importante.

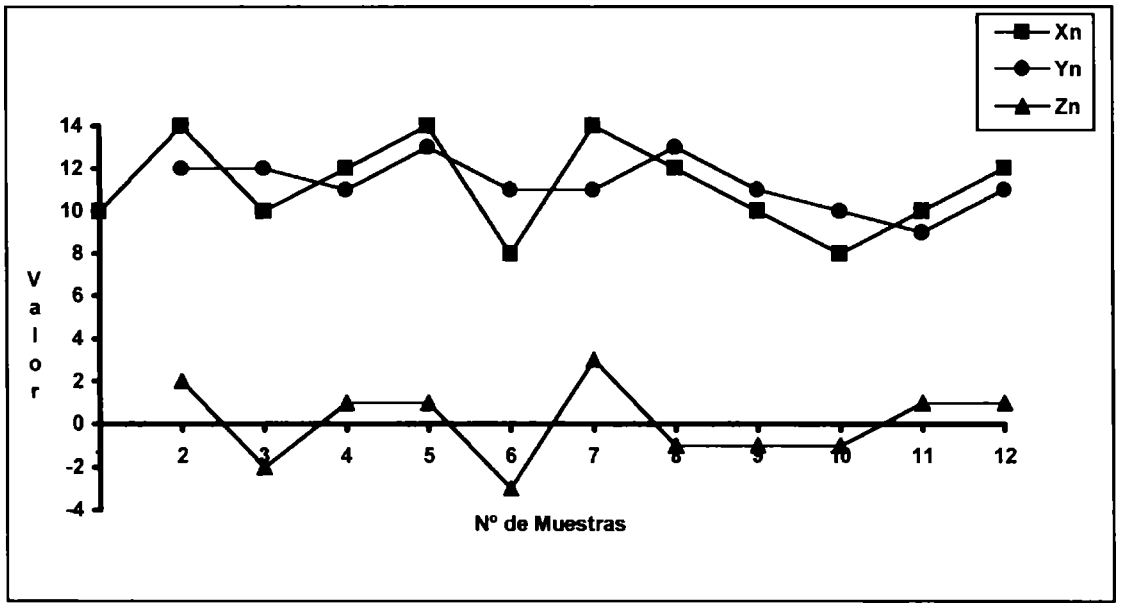


Figura 1 - Conjunto original de muestras y los dos componentes.

Considere la secuencia $\{x_n\}$ ploteada en la Figura 1. Aunque hay una cantidad significativa de variación entre una muestra y la siguiente, hay también una tendencia a largo plazo, mostrada por la línea de puntos, que varía lentamente. Una manera de extraer esta tendencia es hacer el promedio de los valores de las muestras en una ventana movible. La operación de promediar suaviza las variaciones rápidas, haciendo las variaciones lentas más evidentes. Tomemos una ventana de tamaño dos y generemos una nueva secuencia $\{y_n\}$ promediando los valores vecinos de x_n :

$$y_n = \frac{x_n + x_{n-1}}{2}$$

Los valores consecutivos de y_n serán más cercanos unos a otros que los valores consecutivos de x_n. Por lo tanto, la secuencia $\{y_n\}$ puede ser codificada más eficientemente usando codificación diferencial que la secuencia $\{x_n\}$. Sin embargo, queremos codificar la secuencia $\{x_n\}$ no la $\{y_n\}$. Por lo tanto, seguimos a la codificación de la secuencia promediada $\{y_n\}$ de la secuencia diferencia $\{z_n\}$:

$$z_n = x_n - y_n = x_n - \frac{x_n + x_{n-1}}{2} = \frac{x_n - x_{n-1}}{2}$$

Las secuencias $\{y_n\}$ y $\{z_n\}$ pueden ser codificadas independientemente una de otra. De esta manera podemos usar los esquemas de compresión que sean más adecuados a cada secuencia.

Ejemplo :

Supongamos que queremos codificar la siguiente secuencia de valores $\{x_n\}$:

10 14 10 12 14 8 14 12 10 8 10 12

Hay una cantidad significativa de correlación entre una muestra y otra, por lo tanto deberíamos considerar usar un esquema DPCM para comprimir esta secuencia. Para tener una idea de los requerimientos sobre el cuantificador en una esquema DPCM , echemos un vistazo a las diferencias entre muestras sucesivas $\{x_n - x_{n-1}\}$:

10 4 -4 2 2 -6 6 -2 -2 -2 2 2

Ignorando el primer valor, el rango dinámico de las diferencias es de -6 a 6. Supongamos que queremos cuantificar estos valores usando m bits por muestra. Esto significa que podríamos usar un cuantificador con $M=2^m$ niveles o valores de reconstrucción. Si elegimos un cuantificador uniforme, el tamaño de cada intervalo de cuantificación, Δ, es el rango de posibles valores de entrada dividido por el número total de valores de reconstrucción. Por lo tanto, $\Delta = \frac{12}{M}$, el cual nos daría un error máximo de cuantificación de $\Delta/2$ o $6/M$.

Ahora generemos dos nuevas secuencias $\{y_n\}$ y $\{z_n\}$ de acuerdo a las fórmulas de y_n y z_n. Las tres secuencias están ploteadas en la Figura 1. Note que dados y_n y z_n siempre podemos recobrar x_n :

$$x_n = y_n + z_n .$$

Tratemos de codificar cada una de estas secuencias . La secuencia $\{y_n\}$ es

10 12 12 11 13 11 11 13 11 10 9 11

Note que la secuencia $\{y_n\}$ es más suave que la secuencia $\{x_n\}$ — la variación de muestra a muestra es mucho más pequeña. Esto se torna evidente cuando miramos las diferencias entre sucesivas muestras:

10 2 0 -1 2 -2 0 2 -2 -1 -1 2

Las secuencias de diferencias $\{x_n - x_{n-1}\}$ y $\{y_n - y_{n-1}\}$ están ploteadas en la Figura 2.

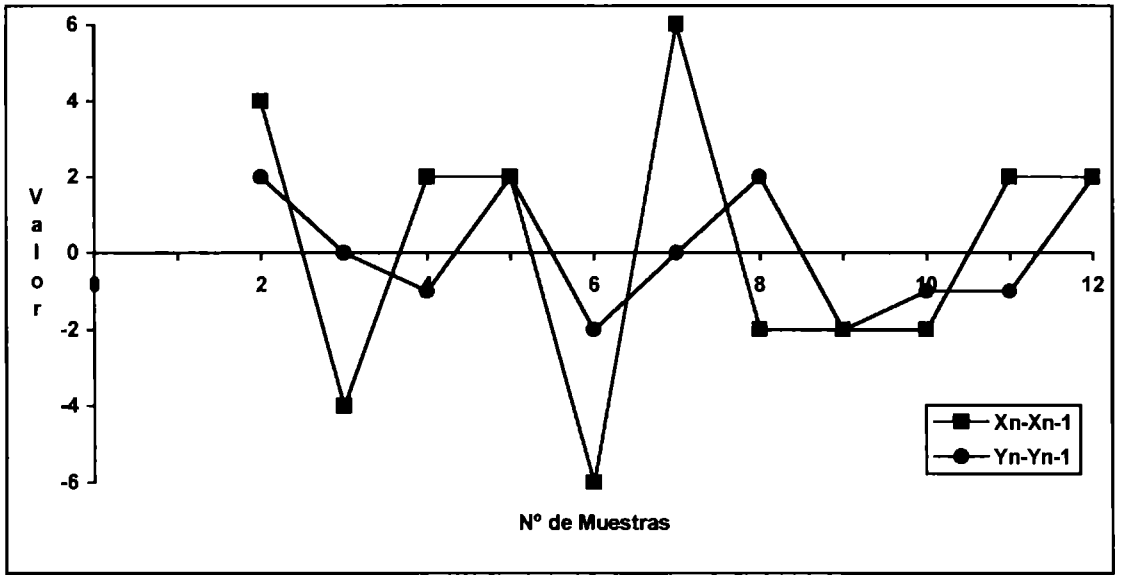


Figura 2 - Diferencia de secuencias generadas entre la secuencia original y promedio.

Nuevamente ignorando la primera diferencia, el rango dinámico de las diferencias es 4. Si tomamos el rango dinámico de esas diferencias como una medida del rango del cuantificador, entonces para un cuantificador de M niveles, el tamaño del paso del cuantificador es $\frac{4}{M}$ y el máximo error de cuantificación es $\frac{2}{M}$. Este máximo error de cuantificación es un tercio del máximo error de cuantificación incurrido cuando la secuencia $\{x_n\}$ es cuantificada usando un cuantificador de M niveles. Sin embargo, para reconstruir $\{x_n\}$ necesitamos transmitir también $\{z_n\}$:

0 2 -2 1 1 -3 3 -1 -1 -1 1 1.

El rango dinámico para z_n es 6, la mitad del rango de la secuencia de diferencias para $\{x_n\}$. Las diferencias entre muestras sucesivas varían más que los valores. Por lo tanto, en lugar de codificar esta secuencia, codificamos cada muestra individual. Para un cuantificador de M niveles, el paso requerido debería ser $6/M$, dando un máximo error de cuantificación de $3/M$.

Para el mismo número de bits por muestra, podemos codificar ambos y_n y z_n e incurrir en menos distorsión. En el receptor sumamos y_n y z_n para obtener la secuencia original x_n. EL máximo error posible de cuantificación en la secuencia reconstruida sería $5/M$, que es menos que el máximo error en que incurriríamos si codificáramos la secuencia $\{x_n\}$ directamente.

Aunque usamos el mismo número de bits por muestra para cada valor de y_n y z_n, el número de elementos en cada una de las secuencias $\{y_n\}$ y $\{z_n\}$ es el mismo que el número de elementos en la secuencia original $\{x_n\}$. Aunque estamos usando el mismo número de bits por muestra, estamos transmitiendo el doble de muestras y, en efecto, doblando la tasa de bits.

Podemos evitar esto enviando valores intercalados de y_n y z_n. Dividamos la secuencia $\{y_n\}$ en subsecuencias $\{y_{2n}\}$ y $\{y_{2n-1}\}$— esto es, una subsecuencia que contenga solamente los elementos numerados impares $\{y_1, y_3 \ldots\}$, y una subsecuencia que contenga solamente los elementos numerados pares $\{y_2, y_4 \ldots\}$. Similarmente, dividimos la secuencia $\{z_n\}$ en subsecuencias $\{z_{2n}\}$ y $\{z_{2n-1}\}$.

Si sólo transmitimos las subsecuencias pares o las impares, transmitiríamos sólo tantos elementos como los de la secuencia original. Para ver cómo recuperamos la secuencia $\{x_n\}$ de estas subsecuencias, supongamos que sólo transmitimos las subsecuencias $\{y_{2n}\}$ y $\{z_{2n}\}$:

$$y_{2n} = \frac{x_{2n} + x_{2n-1}}{2} \qquad z_{2n} = \frac{x_{2n} - x_{2n-1}}{2}$$

Para recuperar los elementos numerados pares de la secuencia $\{x_n\}$, sumamos las dos subsecuencias. Para obtener los miembros impares de la secuencia $\{x_n\}$, tomamos la diferencia:

$$y_{2n} + z_{2n} = x_{2n} \qquad y_{2n} - z_{2n} = x_{2n-1}.$$

Entonces, podemos recuperar la secuencia original completa $\{x_n\}$, enviando sólo tantos bits como los requeridos para transmitir la secuencia original mientras que incurrimos en menor distorsión.

Hay varias cosas que pueden verse de este ejemplo. Primero, el número de diferentes valores que transmitimos es el mismo, tanto en la señal original $\{x_n\}$, o las dos subsecuencias $\{y_{2n}\}$ y $\{z_{2n}\}$. Descomponiendo la secuencia $\{y_{2n}\}$ y $\{z_{2n}\}$ en subsecuencias no resultó en un incremento del número de valores que necesitamos transmitir. Segundo, las dos subsecuencias tenían distintas características, lo cual lleva al uso de diferentes técnicas para codificar las diferentes secuencias. Si no hubiéramos partido la secuencia $\{x_n\}$, hubiéramos estado usando esencialmente el mismo enfoque para codificar las dos subsecuencias. Finalmente, podríamos haber usado el mismo enfoque para descomponer las dos secuencias constituyentes, que entonces podrían ser descompuestas aún más.

2.1- Dominio frecuencial y filtrado

En el ejemplo anterior, separamos una secuencia $\{x_n\}$ en dos componentes: una componente con menores diferencias entre muestras sucesivas y una componente con mayores diferencias entre muestras sucesivas. Si viéramos $\{x_n\}$ como muestras de una señal que varía en el tiempo, podríamos decir que descompusimos la señal en una componente da cambios lentos y una componente de cambios rápidos.

Para poder hablar más precisamente sobre señales con diferentes tasas de cambio, necesitamos tener alguna medida de la tasa de cambio. Una medida de cuan rápido cambia una señal con respecto al tiempo es la frecuencia de la señal. Una señal de alta frecuencia cambia más rápidamente que una señal de baja frecuencia. La frecuencia como medida de cambio con respecto al tiempo es expresada en unidades de Hertz(Hz). Una señal sinusoidal que completa 60 ciclos por segundo tiene una frecuencia de 60 Hz.

Siempre que hablamos de descomponer una señal en sus componentes frecuenciales, pensamos en términos de sinusoides, pero éste no es siempre el caso. Podemos descomponer cualquier señal en sinusoides tanto de valores reales o complejos. Si una señal f(t) es periódica de período T, podemos representarla en términos de sinusoides como sigue:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(n \tfrac{2\pi}{T} t\right) - \sum_{n=1}^{\infty} b_n \operatorname{sen}\left(n \tfrac{2\pi}{T} t\right).$$

Esta representación es llamada representación en series de Fourier. Si f(t) es una función par (f(-t)=f(t)), todos los valores b_n son cero porque la función seno es par. Similarmente si f(t) es una función impar (f(-t)=-f(t)), todos los valores an son cero porque la función coseno es una función impar.

Las series de Fourier también pueden ser escritas en términos de exponenciales complejos como

$$f(t) = \sum_{n=-\infty}^{\infty} Cne^{jn\frac{2\pi}{T}t}$$

El concepto de frecuencia también puede ser usado para medir el cambio en el valor de una señal en el espacio. Esto se torna necesario con las señales bidimensionales tales como las imágenes. Cuando hablamos de frecuencia en el contexto de las

imágenes, generalmente nos referimos a tasas de cambio en las direcciones horizontal y vertical. Estas tasas de cambio son medidas por una frecuencia espacial horizontal y vertical.

La mayoría de las señales de interés están compuestas por un amplio rango de frecuencias. La voz humana consiste de componentes frecuenciales que van desde alrededor de 300Hz. hasta alrededor de 10 kHz. Las señales de audio como piezas de orquesta y música de rock and roll pueden contener componentes tan grandes como 20 kHz. Las imágenes que contienen mucho detalle tienen mayores frecuencias espaciales, mientras que las imágenes que tienen solamente variaciones suaves de intensidad contienen solamente bajas frecuencias espaciales. Por lo tanto si deseamos codificar una señal que contiene tanto componentes de alta como de baja frecuencia, es útil separarla en dos señales: una que contenga las frecuencias más altas y otra que consista en frecuencias más bajas.

Esta descomposición es útil desde otro punto de vista también. Para señales que serán finalmente percibidas por humanos, esta separación en componentes frecuenciales nos permite usar la naturaleza selectiva por frecuencias de la percepción humana para guiar nuestras técnicas de codificación.

En el codificador sub-banda, la banda de la señal es dividida típicamente en cuatro o más sub-bandas por un conjunto de filtros pasa banda. Cada sub-banda es luego remuestreada a su tasa de Nyquist y digitalmente codificada. En este proceso cada banda puede ser codificada de acuerdo al criterio perceptual específico de esa banda. En la reconstrucción, las señales de sub-banda son decodificadas y entonces sumadas para dar una réplica cercana a la señal original.

La división en sub-bandas se ilustra en la Figura 3; el caso especial de sub-bandas de igual ancho es importante para implementación.

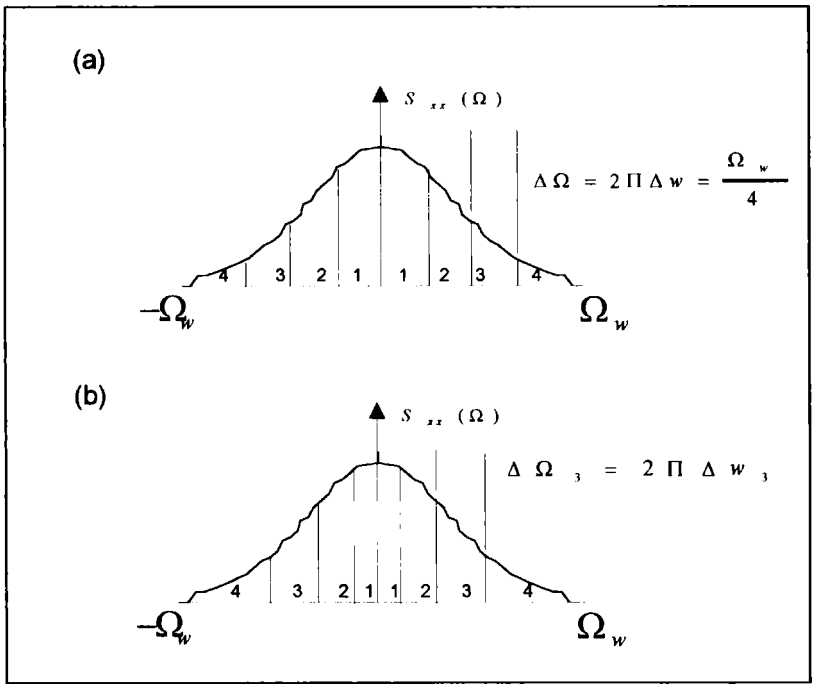


Figura 3 - División del espectro de entrada en M=4 sub-bandas de ancho constante (a) y variable (b).

La codificación en sub-bandas ofrece varias ventajas. Alocando apropiadamente los bits en las diferentes bandas, el número de niveles de cuantificación y por lo tanto la varianza del error de reconstrucción pueden ser separadamente controlados en cada banda, y la forma del espectro del error total de reconstrucción puede controlarse como una función de la frecuencia. En las frecuencias más bajas, donde la estructura debe ser exactamente preservada, un mayor número de bits por muestra pueden ser usados; mientras que en las frecuencias más altas, donde es más frecuente que aparezca el ruido, menor cantidad de bits por muestra pueden ser usados.

2.2- Filtros

La separación de estos componentes es llamada filtrado. Un sistema que aísla ciertos componentes frecuenciales es llamado filtro. La analogía aquí con los filtros mecánicos tales como el filtro de café es obvia. Un filtro de café o un filtro en un sistema purificador de agua bloquea las partículas gruesas y deja pasar solo los componentes más finos. La analogía no es completa, sin embargo, pues los filtros mecánicos siempre bloquean los componentes gruesos de la entrada, mientras que los filtros que aquí se tratan pueden selectivamente dejar pasar cualquier rango de

frecuencias.. Los filtros que solamente dejan pasar componentes por debajo de cierta frecuencia f_0 son llamados filtros pasa bajo; los filtros que bloquean todos los componentes frecuenciales por debajo de cero valor f_0 son llamados filtros pasa alto. La frecuencia f_0 es llamada frecuencia de corte. Los filtros que dejan pasar componentes que tienen contenido frecuencial por sobre cierta frecuencia f_1 pero por debajo de alguna frecuencia f_2 son llamados filtros pasa banda.

Una manera de caracterizar a los filtros es por su función de transferencia de magnitud — el cociente de la magnitud de la entrada y la salida del filtro como una función de la frecuencia. En la Figura 4 se muestra la función de transferencia de magnitud para un filtro pasa bajo ideal y para un filtro pasa bajo más realista, ambos con frecuencia de corte en f_0. En el caso ideal, ningún componente de la señal de entrada por debajo de f_0 es afectado excepto por una cantidad constante de amplificación. Todas las frecuencias por sobre f_0 son bloqueadas. En otras palabras, el corte es abrupto. En el caso de un filtro más realista, el corte es más gradual. También, la amplificación para los componentes con frecuencia menor que f_0 no es constante, y los componentes con frecuencias mayores que f_0 no son totalmente bloqueados. Este fenómeno se conoce como rebote en la banda que pasa y en la que queda.

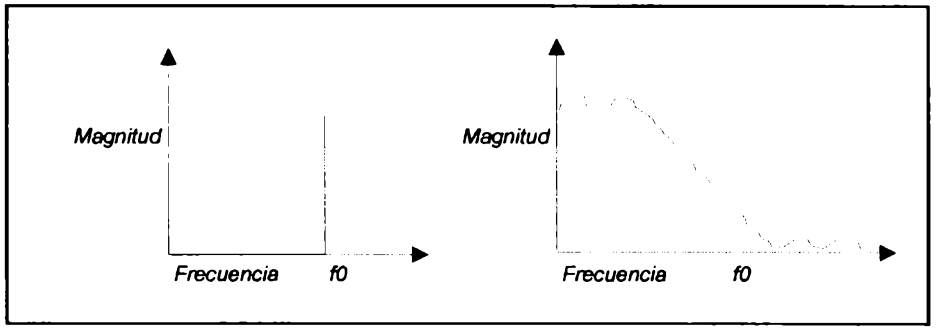


Figura 4 - Características de filtros pasabajos ideal y real.

Los filtros que tratamos son digitales; operan sobre una secuencia de números que son usualmente muestras de una señal continua.

2.2.1- Conjunto de filtros de banda entera

Una importante característica en los filtros es que las frecuencias de corte sean elegidas tal que cada banda pueda ser muestreada a dos veces el ancho de banda correspondiente, en lugar de dos veces la frecuencia más alta de la señal de banda completa. Esto es posible en la situación especial de los filtros de muestreo de banda entera, donde al frecuencia de corte f_{kl} en una sub-banda es múltiplo entero del ancho de banda:

$f_{kl} = n\Delta f_l$ *n=0,1,2,...* *l=1,2,3,...,M* *M: nro. de bandas*

Considere una entrada discreta x(n), muestreada a la tasa de Nyquist de la señal de banda completa $f_s=2W$ donde $W = \sum \Delta f_l$, con la suma sobre l=1 a M, es la frecuencia máxima de la señal de banda completa.

Sean las sub-bandas escritas en la forma

$\Delta f_l = W/\zeta$; k=1,2,...,M

$\zeta = M$ *para todos los l con sub-bandas de igual ancho*

La Figura 5 muestra la secuencia de operaciones de filtrado y codificación en codificación sub-bandas. El decimador ζ:1 submuestrea la salida del filtro xk(n) en un factor de ζ,implicando una tasa de muestreo de $f_{sk} = 2W/\zeta = 2\Delta f_k$ para cada sub-banda k. El interpolador 1:ζ completa con ceros entre cada par de muestras.

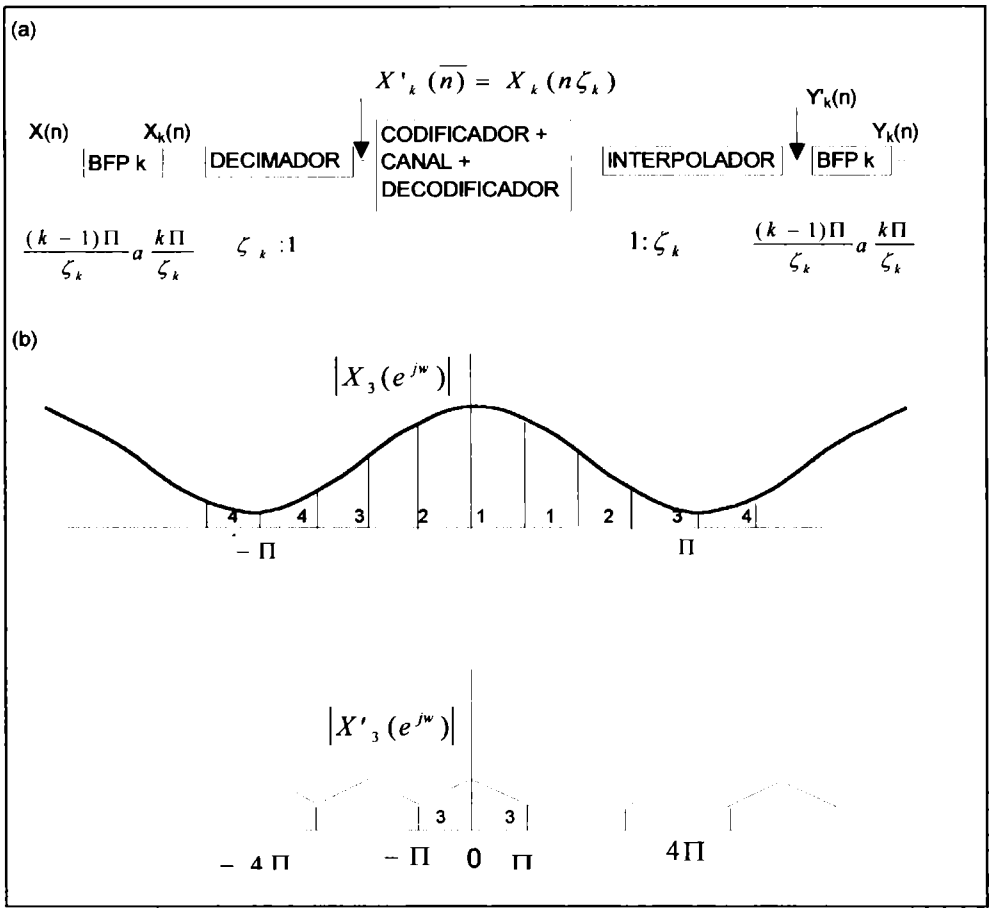


Figura 5 - Realización de muestreo de banda entera con una entrada discreta. (a) diagrama de SBC para k sub-bandas ; (b) espectro original y remuestreado luego de la decimación.

2.2.2- Conjunto de filtros espejo en cuadratura

La situación de superposición de sub-bandas sugiere que el aliasing puede ocurrir si las sub-bandas son muestreadas exactamente al doble del ancho de banda correspondiente. Este problema es elegantemente abordado en el conjunto de filtros espejo en cuadratura(QMF), de la Figura 6. Esta figura muestra la división de una señal de banda completa de máxima frecuencia en radianes ¶ en dos de igual ancho usando un par de filtros pasa bajo y pasa alto. Note que los filtros son simétricos:

$$h_{n-1-n} = h_n \qquad n = 0,1,\ldots,\tfrac{N}{2} - 1$$

Cada una de las señales $x_l(n)$ y $x_u(n)$ es remuestreada en factor de 2:1. Esta reducción de las tasas de muestreo de sub-banda es necesaria para mantener una tasa total mínima de bits en la codificación de estas señales. En el proceso de reconstrucción , las tasas de muestreo son incrementadas en un factor de 1:2 completando con ceros entre cada par de muestras de sub-banda.

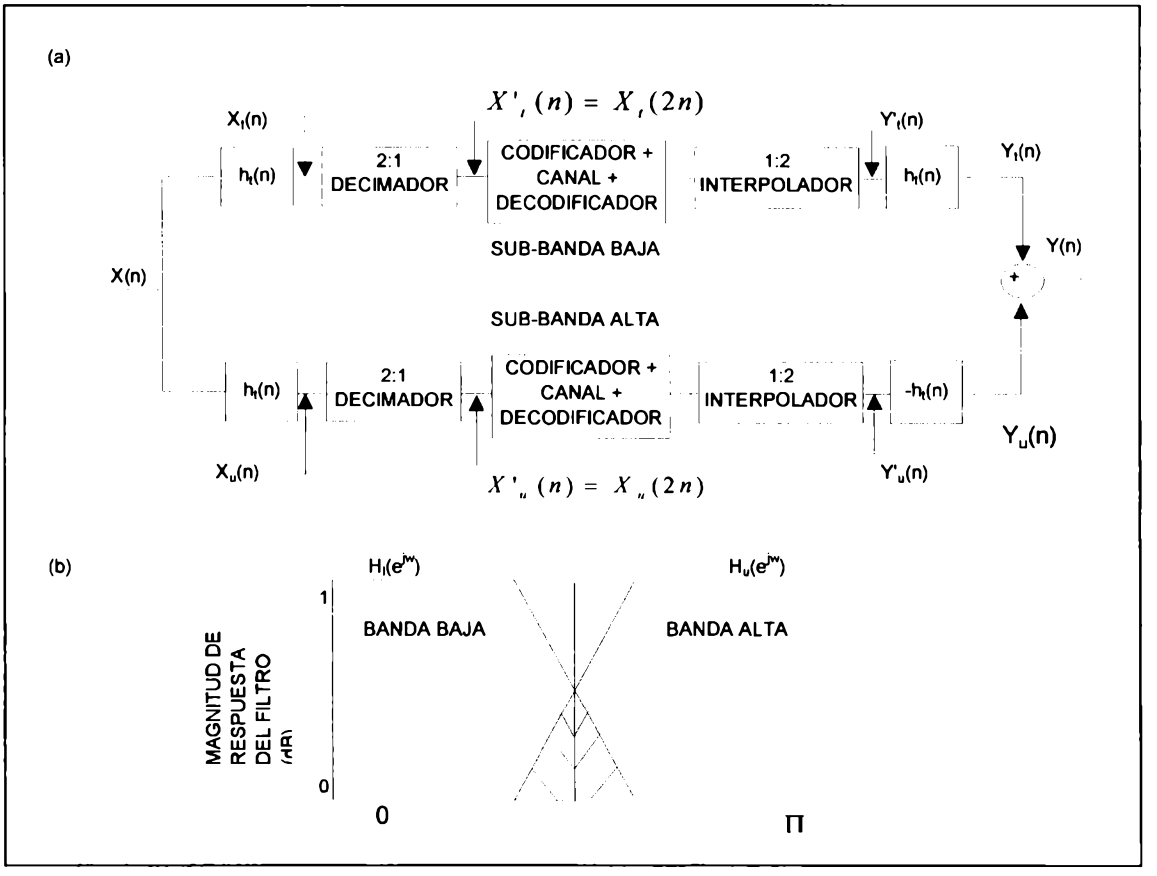


Figura 6 - Filtrado en espejo en cuadratura dividiendo la entrada en dos sub-bandas de igual ancho. (a) implementación ; (b) muestra la cancelación con aliasing.

Tanto la decimación como la interpolación introducen ruido a la señal reconstruida, en términos de aliasing y de repetición periódica respectivamente. El grado en que este ruido es removido por los filtros es determinado por el grado en que éstos se aproximan a su correspondiente ideal. Por la relación especial de las señales sub-bandas en el conjunto QMF, los componentes que quedan por repetición periódica pueden ser cancelados por los términos de aliasing introducidos en el análisis. Esta cancelación ocurre después de la suma de las señales filtradas e interpoladas, y la cancelación es exacta en ausencia de error de codificación. En presencia de codificación, esta cancelación es obtenida de acuerdo al nivel de ruido de cuantificación.

2.3- Algoritmo básico de codificación sub-banda

El sistema básico de codificación sub-banda es mostrado en la Figura 7. La salida de la fuente es pasada por un conjunto de filtros , llamado conjunto de filtros de análisis, el cual cubre el rango de frecuencias que componen la salida de la fuente. Las bandas que pasan de los filtros pueden o no superponerse. Las salidas de los filtros son entonces sub-muestreadas. Una vez que las salidas de los filtros ha sido decimada, la salida es codificada usando uno o varios esquemas.

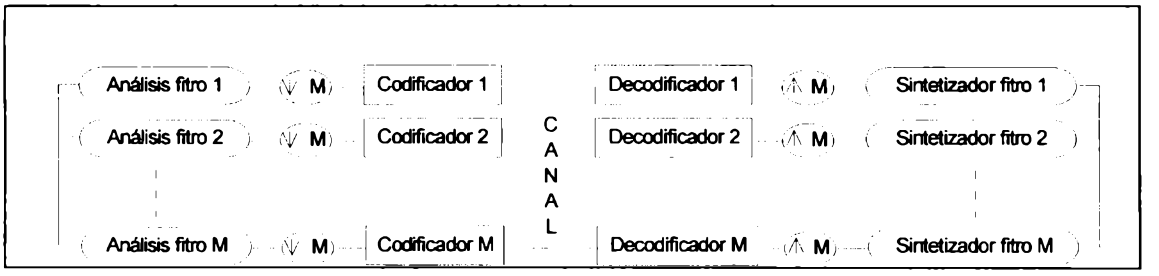


Figura 7 - Diagrama de un Sistema de Codificación sub-banda.

2.4- Aplicación a compresión de imágenes

¿Cómo hacer cuando las señales contienen dependencias bidimensionales como en el caso de las imágenes? La respuesta obvia es que necesitamos filtros de dos dimensiones, que separen la salida de la fuente en componentes basados tanto en frecuencias horizontales como verticales. Afortunadamente, en la mayoría de los casos, este filtro bidimensional puede ser implementado como dos filtros unidimensionales, los que puedan ser aplicados primero en una dimensión y luego en la otra. Los filtros que tienen esta propiedad se llaman separables.

Generalmente, para codificación sub-banda de imágenes cada fila de la imagen es filtrada separadamente usando un filtro pasa alto y uno pasa bajo. La salida de los filtros es decimada en un factor de dos. Suponga que la imagen es de tamaño N x N. Después del primer paso, tendremos dos imágenes de tamaño N x N/2. Entonces filtramos cada columna de las dos subimágenes, decimando nuevamente las salidas por un factor de dos. Esto resulta en cuatro imágenes de tamaño N/2 x N/2. Podemos parar en este punto o continuar el proceso de descomposición con una o más de las cuatro subimágenes. Generalmente de las cuatro subimágenes sólo una o dos se siguen descomponiendo. La razón es que muchos de los valores de los

pixels en las subimágenes de alta frecuencia son cercanos a cero, por lo cual no vale la pena gastar tiempo computacional en descomponer esas subimágenes.

3- WAVELETS

3.1- introducción

3.1.1- Pirámides de imagen

Las pirámides pueden ser usadas para el filtrado de imágenes, construyendo valores a partir de la señal original.

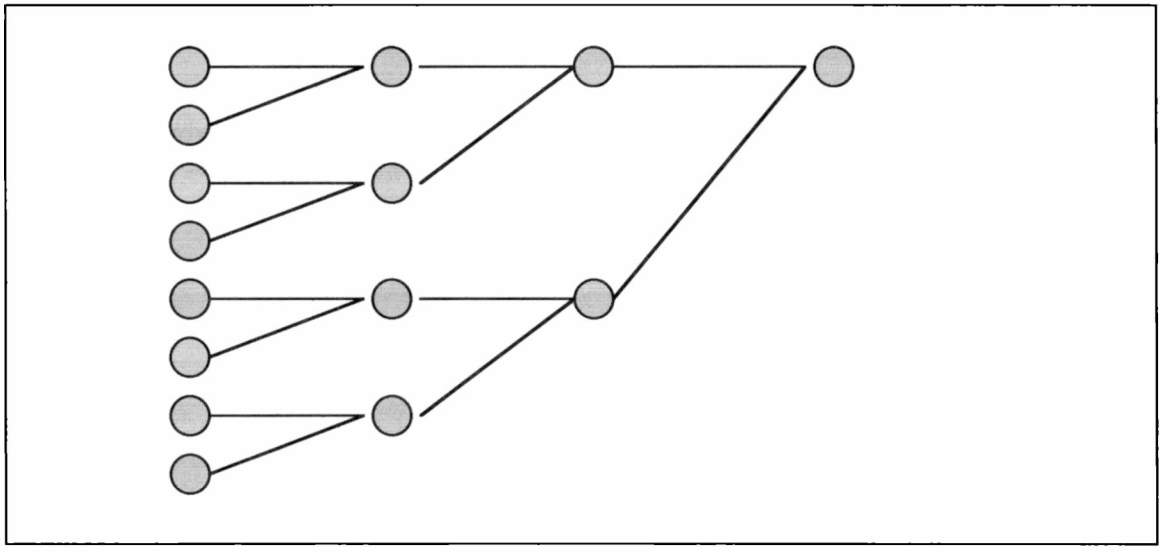


Figura 8

Esta pirámide puede verse como el resultado de aplicar filtros a escala sobre la señal. Para *n* valores iniciales, tenemos $\log_2(n)$ pasos y $2n$-1 nodos. Sólo se tuvieron que computar n-1 sumas. Sin embargo este no es un buen esquema para reconstrucción, pues hay redundancia en los datos. Llamando $s_{i,j}$ al j-ésimo elemento del nivel y (0 es el tope de la pirámide, $k=\log_2(n)$ el nivel más bajo) tenemos:

$$s_{i,j} = \frac{\left(s_{i+1,2j} + s_{i+1,2j+1}\right)}{2}$$

Podemos ahora almacenar $s_{0,0}$ como antes pero en el nivel de abajo almacenamos:

$$s'_{1,0} = \frac{\left(s_{1,0} - s_{1,1}\right)}{2}$$

Es claro que sumando $s_{0,0}$ y $s'_{1,0}$ recobramos $s_{1,0}$ y restando $s_{0,0}$ y $s'_{1,0}$ recobramos s1,1. Por lo tanto tenemos la misma información con un elemento menos. La misma modificación aplicada recursivamente a través de la pirámide resulta en n-1 valores almacenados en k-1 niveles. Como necesitamos el valor tope así como $s_{0,0}$, y las

sumas como resultados intermedios, el esquema computacional ahora es el siguiente:

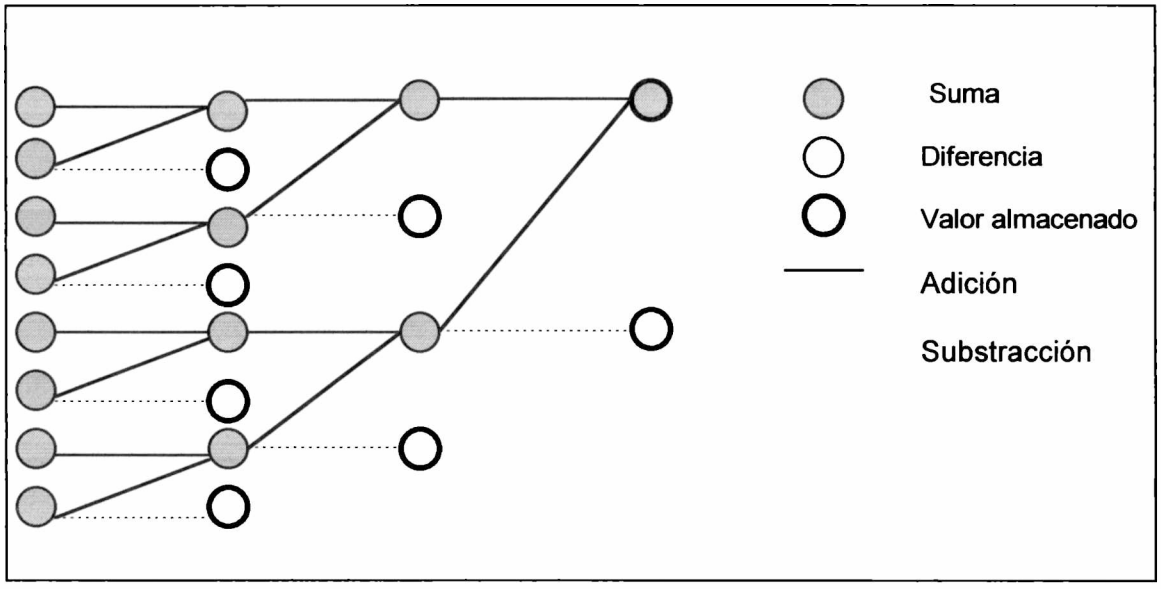


Figura 9

Ahora podemos generalizar el concepto de pirámides de imagen a lo que se conoce como esquema de codificación subbanda. Se aplican recursivamente dos operadores a la señal para submuestrearla por 2. El primero es la función caja, y es un suavizado, o un filtro pasabajo, y el otro es la wavelet básica de Haar, o detalle o un filtro pasaalto. En general, si tenemos un filtro pasabajo h(n), un filtro pasaalto g(n), y una señal f(n), podemos computar la versión suavizada y submuestreada:

$$a(k) = \sum_i f(i)h(-i+2k)$$

y la versión detallada submuestreada:

$$d(k) = \sum_i f(i)g(-i+2k)$$

Si el filtro de suavizado es ortogonal a su traspuesto, entonces los dos filtros están relacionados por:

$$g(L-1-i) = (-1)^i h(i)$$

(donde L es la longitud del filtro). La reconstrucción es entonces exacta , y es computada como:

$$f(i) = \sum_k [a(k)h(-i+2k) + d(k)g(-i+2k)]$$

Podemos aplicar este esquema recursivamente a la nueva señal suavizada, que es de la mitad del tamaño de f(), hasta que tenga dos vectores a() y d() de longitud 1 después de $\log_2(n)$ aplicaciones.

El esquema computacional es el siguiente:

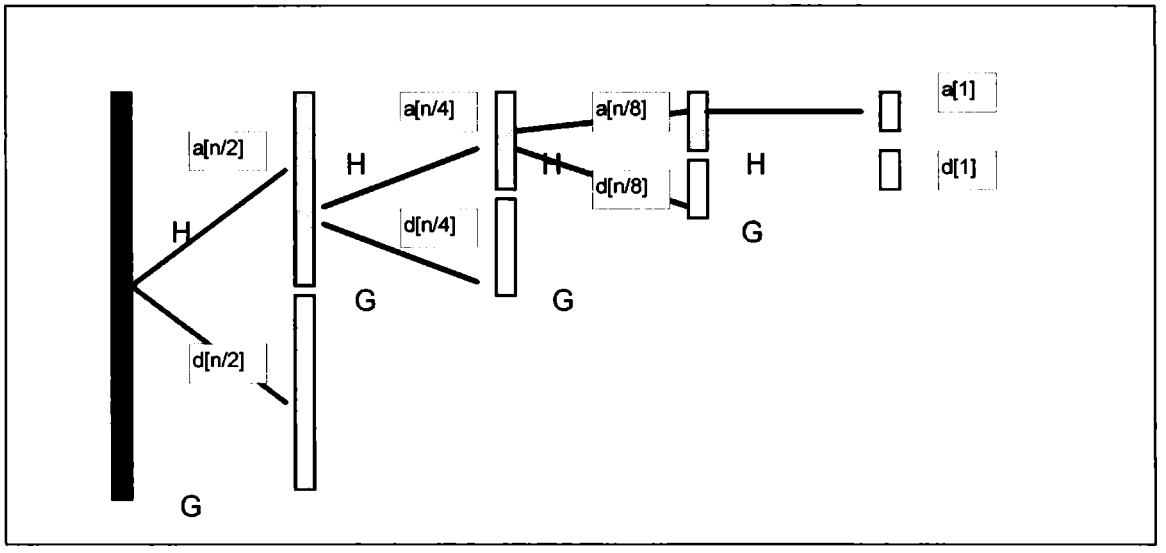


Figura 10

3.2- Transformada de Fourier

La trasformada wavelet es una proyección de una señal en una serie de funciones bases llamadas *bases wavelet.*

El motivo por el que se estudia las wavelets, es que para algunos análisis de señales, son más convenientes que la transformada de Fourier. La mayoría de las señales que tratamos en una computadora son *no estacionarias*, esto es, no son estadísticamente la misma en cualquier lugar. En nuestro caso forzamos, mucha información de alta frecuencia en algunos lugares y poca en otros. Para ello se usan técnicas de supermuestreo adaptativas. En las regiones donde la imagen tiene muchos bordes, texturas y sombras, necesitamos muestrear más veces para determinar las altas frecuencias de la región. Si la imagen tiene pocos cambios de color, podemos muestrear en forma más espaciada.

La única herramienta analítica que tenemos para analizar la frecuencia contenida en una señal es la trasformada de Fourier.

La trasformada de Fourier no permite observar la frecuencia en un intervalo de la señal independiente del resto ; siempre integra sobre la señal completa retornando

luego al intervalo deseado. Si se encuentra una ráfaga de componentes de alta frecuencia en el medio de un conjunto de componentes predominantes de baja frecuencia, la transformada de Fourier no nos dice nada acerca de la ubicación de la ráfaga.

El deseo de describir el contenido frecuencial en un segmento corto de una señal, nos conduce al desarrollo de una variedad de técnicas que proveen descripciones locales de señales (como la transformada de Fourier Short-Time).

Las *wavelets* fueron desarrolladas para solucionar ese problema. La transformada wavelet toma una señal de entrada y la proyecta en un nuevo conjunto de funciones base llamadas *wavelets*. Una característica sobresaliente es que las wavelets pueden tener soporte compacto. Para hacer match en una señal finita, podemos poner junto un número finito de wavelets finitas. Las wavelets son combinadas con sus parámetros apropiados para reconstruir la señal de entrada.

El problema de la localización de rangos de frecuencias, son fácilmente solucionadas con las wavelets. El análisis de las wavelets nos permite obtener una señal con diferentes resoluciones (número de muestras en una señal - nivel de detalle).

Las wavelets forman una familia 2D de funciones derivadas de la función original, *v*, llamada *función de escala*. A partir de la función de escala, creamos una *wavelet madre*, y todas las otras wavelets surgen del ella en una versión a escala, dilatada y corrida. Las wavelets pueden formar una base ortonormal y puede ser implementada rápidamente usando la *transformada wavelet rápida* análogamente a la *transformada rápida de Fourier*.

3.2.1- Transformada de Fourier Short-Time (por ventanas)

Recordemos que la transformada de Fourier está definida sobre todo el dominio de la señal, y si se truncan algunos términos se ve afectada la calidad de toda la señal.

En una imagen el contenido de alta frecuencia de un background suave es pequeño, pero si hay un objeto de textura compleja en el foreground se esperará que el contenido de alta frecuencia en esa región sea mayor. Es natural aislar secciones de la señal y tomar transformadas de Fourier independientes sobre esos trozos.

Este enfoque es llamado *transformada de Fourier por ventanas* (STFT).

Para tomar la STFT de una señal, multiplicamos la señal por una función ventana que es típicamente distinta de cero en la región en la que estamos interesados. Esta función ventana es central a este método.

La clave de la STFT es encontrar una función ventana que simultáneamente aísle solo la parte de la señal en la que estamos interesados (sin afectar la señal), elimine el resto de la señal y no introduzca ruido (esto usualmente implica una transición suave en los bordes). Estos objetivos son mutuamente antagónicos.

3.3- Escala y Resolución

La transformada de Fourier principalmente se ocupa de la información de *frecuencia* y *fase* de la señal. La transformada wavelet en alguna manera es similar y además nos permite observar una señal desde diferentes *escalas* y *resoluciones*.

Lo anteriormente mencionado tiene un significado mayor en procesamiento de imágenes ; nos será útil conocerlo antes introducirnos más con las wavelets.

Veremos primero *escala*. Imaginemos un mapa de una ciudad, donde 1 cm. es igual a 1 km. Si reducimos el mapa en la mitad, 1 cm. corresponderá a 2km ; la escala se ha duplicado. Similarmente, si agrandamos el doble al mapa original, la escala se ha dividido por 2. Entonces describe cuánto se corresponde la señal original en una unidad (lo cual es generalmente el valor asociado con una muestra igualmente espaciada) de la representación de la señal.

Cuando se aplica en wavelets, *resolución* se refiere a la suma de información de una señal ; esto está relacionado a sus contenidos frecuenciales. Si filtramos los componentes de baja frecuencia, no debemos cambiar la escala pues se altera su resolución.

Para ejemplificar esto, supongamos que tenemos una señal discreta de cuatro elementos, s0 =[0,0,2,2]. Podemos crear una nueva versión de la señal promediando pares sucesivos de valores, creando s1=[0,2]. Como el mapa se ha achicado, cada elemento de s1 cubre dos veces el territorio de la señal original, por lo tanto la escala se ha duplicado. Si repetimos la operación obtendremos s2=[1] ; la escala de s2 es cuatro veces la escala de s0 y la resolución ha cambiado. Supongamos que volvemos a la señal original s0, y filtramos los componentes de baja frecuencia, creando sf=[0.25,0.75,1.25,1.75]. La escala de sf es la misma que s0, pero la resolución de la señal ha cambiado.

La idea básica es crear distintas señales desde la original, cada una a diferente escala. Las wavelets nos permiten ejecutar el equivalente a una reducción de un mapa, construyendo señales pequeñas que contienen la misma información general que las señales más grandes.

3.4- La Ecuación de Dilación y la Transformada de Haar

Un ejemplo canónico de bases wavelets, es la *wavelet Haar.*

El centro de la wavelet es la *ecuación de dilación.* Nos dice cómo crear una función desde versiones dilatadas, a escala y corridas de la original (función canónica). Para una función dada *v(x)*, podemos dilatar (ej : comprimir o agrandar) esa función computando *v(ax)*, para algún *a*, podemos hacer escala la función computando *sv(x)*, para algún *s*, y podemos correr la función computando *v(x+b)*, para algún *b*. Combinándolos tendremos *sv(ax+b)*. Para una dilación con factor de 2, la ecuación de dilación será :

$$v(x) = \sum_{k=0}^{N} c_k \, v(2x - k)$$

Un simple ejemplo de una función que satisface la ecuación de dilación es una caja *y0(t)* sobre el intervalo [0,1].

y0(t)= 1, si 0<=*t*<=1 ; 0, caso contrario.

En la función de dilación inicializaremos todos los coeficientes a 0 excepto *c0*=*c1*=1. Entonces :

y1(t)= y0(2t)+ y0(2t-1)

y1(t) comprende de dos cajas de la mitad del ancho de *y0*, de la misma forma obtenemos *y2* que serán dos copias de *y1*, es decir, cuatro copias de *y0*. Podemos continuar la recursión en forma indefinida.

(Ver figura 11)

La función de dilación nos dice cómo encontrar una función *v* que pueda construirse de la suma de copias de si misma. Una función que satisface este criterio es llamada *función de escala.* Para cada función de escala *v* podemos crear el conjunto correspondiente de *wavelets*, la cual son las funciones base de la transformada wavelet.

Para construir una wavelet a partir de la función de escala es similar a la ecuación de dilación, usa los mismos coeficientes en orden contrario y alternando los signos. Para construir la primera wavelet (w0), aplicamos :

$$w0(t) = \sum_{k \in Z} (-1)^k c_{1-k} v(2t - k)$$

Las wavelets pueden ser escritas como $w^{a,b}$, la cual están basadas en la wavelet original w0 (a esta se la llama wavelet madre o wavelet Haar). Los parámetros a y b tienen el siguiente significado en la fórmula :

$$w^{a,b}(t) = \frac{1}{\sqrt{a}} w0\left(\frac{t-b}{a}\right)$$

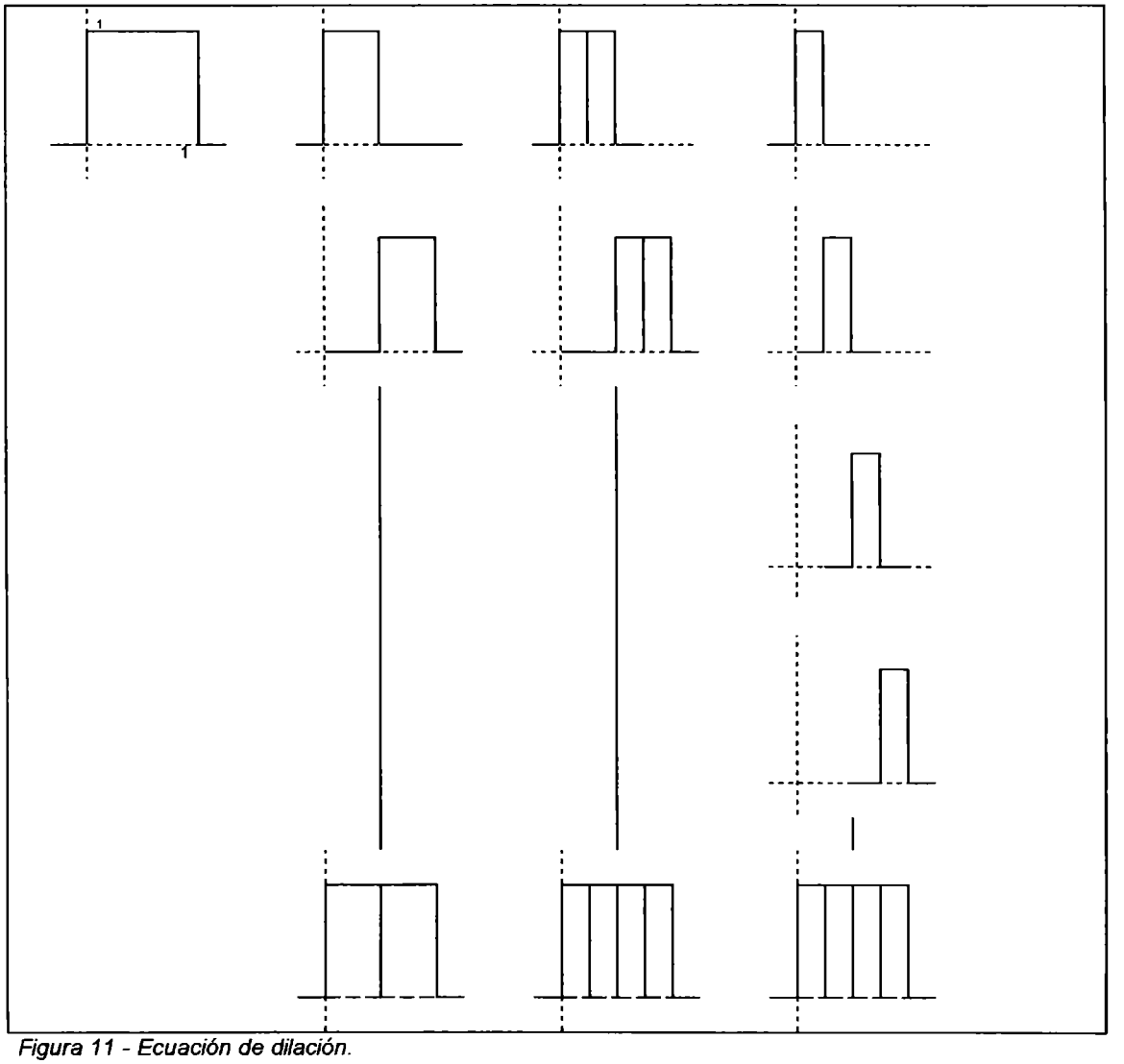


Figura 11 - Ecuación de dilación.

3.5- Descomposición y Reconstrucción

La transformación wavelet se construye a partir de la función de escala *v(t)*, los coeficientes wavelet $b^{a,b}$, y las wavelets $w^{a,b}(t)$.

Para ver como trabaja, consideremos una función *x(t)* constante a trozos en ¼ de una unidad de intervalo.

Figura 12 - Función constante a trozos en una unidad de intervalo.

La función de escala *v(t)* y las primeras dos generaciones de wavelets, $w^{0,0}$ a $w^{1,1}$, se muestran a continuación. Supongamos que tenemos un vector de entrada de cuatro valores, $x(t)= 3v(t) + -2w^{0,0}(t) + 4w^{1,0}(t) + -3w^{1,1}(t)$ *(1)*

Podemos ver cómo contribuyen con el total :

	3	3	3	3	$3v(t)$
+	-2	-2	2	2	$-2w^{0,0}(t)$
+	4	-4	0	0	$4w^{1,0}(t)$
+	0	0	-3	3	$-3w^{1,1}(t)$
=	5	-3	2	8	$x(t)$

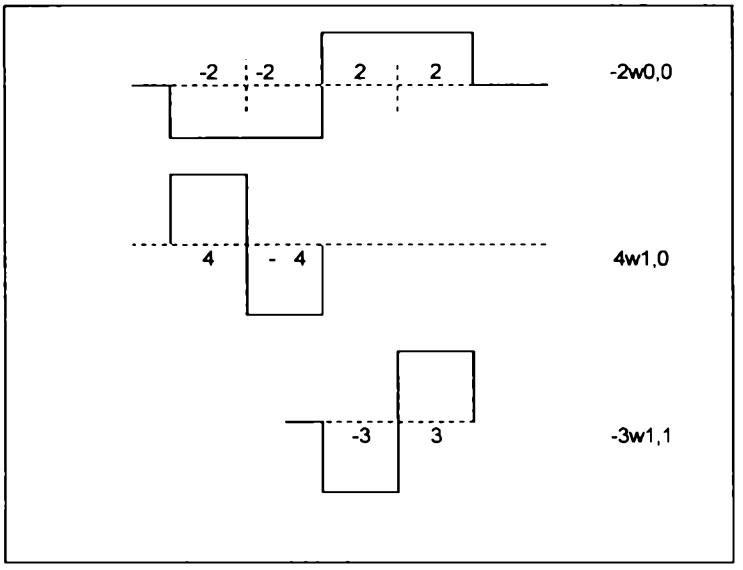


Figura 13 - Las primeras dos generaciones de wavelets.

Los números (3,-2,4,-3) en *(1)* son los coeficientes de las funciones wavelet que hacen match con la función de entrada.

Los índices *a* y *b* de los coeficientes se refieren al nivel de la wavelet y su posición. El índice *a* nos dice cuánto se dilató la wavelet. Los niveles altos de *a* indican wavelet con bases afinadas, y son capaces de capturar cambios de información rápidamente. Los valores de *b* nos dicen dónde se aloca la wavelet en la señal. Cada vez que incrementamos la resolución, duplicamos el número de wavelets, en general para algún valor de *a* y *b* será de 0 a 2^a-1.

La idea básica de las wavelets es filtrar la señal original con un filtro pasabajos para obtener una versión 'suavizada', y luego encontrar las diferencias entre ésta y la original. Luego, suavizamos otra vez y hallamos las diferencias entre la nueva y la anterior, así continuar hasta que la señal se haya suavizado en un número simple.

El siguiente ejemplo se basa en la señal $x=[5,-3,2,8]^t$.

En la transformada wavelet de Haar, la amplitud de la función de escala *v*, denotada por b^v, representa el valor promedio de la señal, como se muestra en la siguiente figura :

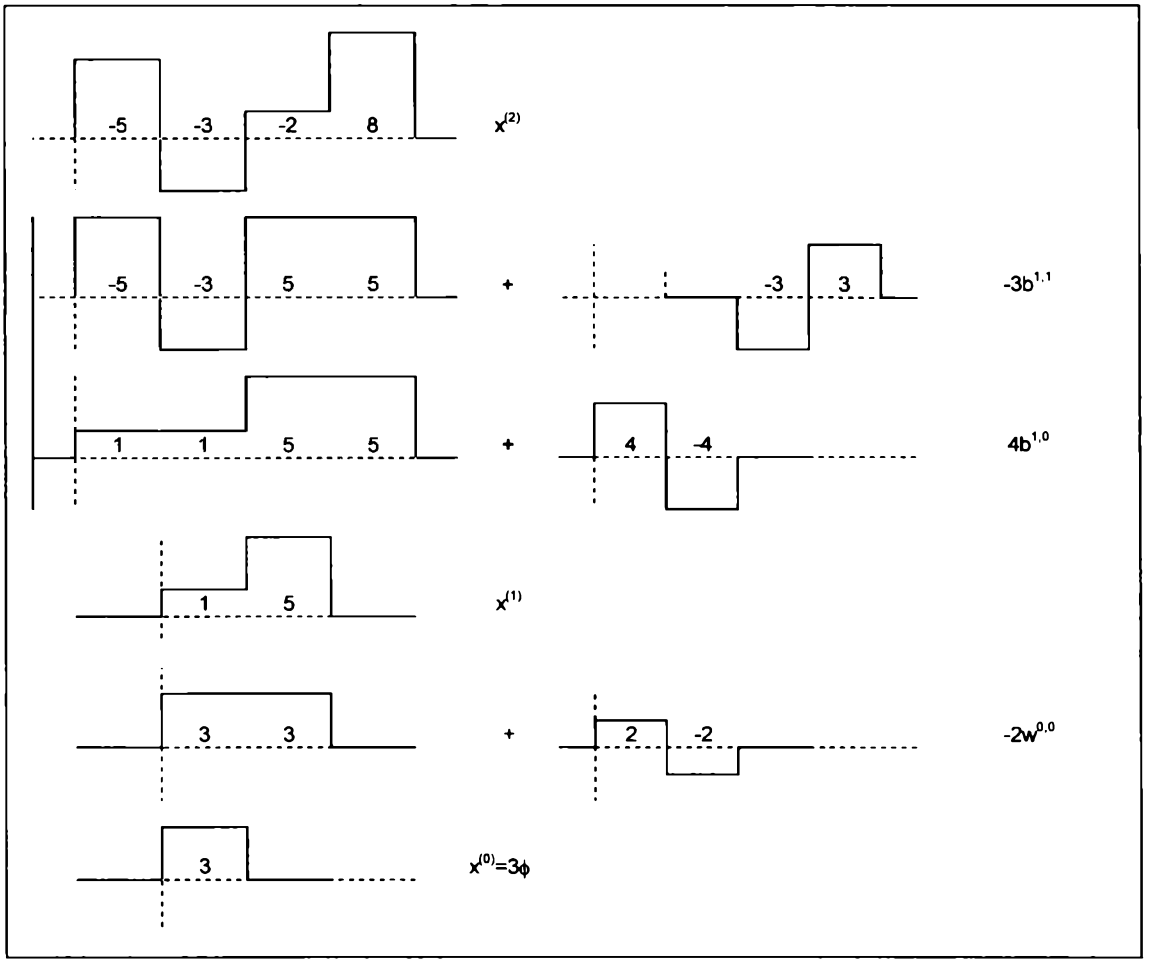


Figura 14 - La señal de entrada x=[5,-3,2,8]'. Debajo, las series de wavelets.

Comenzando desde el último gráfico y subiéndonos en la figura, para construir los niveles superiores, primero sumamos la wavelet $w^{0,0}$ con escala en $b^{0,0}$,con la versión supermuestreada de la función $x^{(0)}$, la cual divide la señal en dos piezas constantes. Las próximas wavelets son cada una de medio intervalo de ancho, por lo tanto a cada una se le hace escala en $b^{1,0}$ y $b^{1,1}$.

Para computar los coeficientes se transforma la señal de arriba hacia abajo. En cada paso se estará suavizando la señal pasando un simple filtro pasabajos sobre ella.

La transformada de Haar repite un simple paso muchas veces. Se toman dos valores adyacentes en la señal y se lo reemplaza por dos nuevos números : su promedio y la amplitud de la wavelet que se le sumará al promedio para recrear los valores

originales. En cada paso reemplazamos un par de valores p y q por su media m y la amplitud de la media w. Dado un par de valores adyacentes p y q :

$$m = (p + q) / 2 \qquad w = (p - q) / 2$$

Para recuperar p y q :

$$p = m + w \qquad q = m - w$$

Los valores w son los coeficientes wavelets de esa sección de la señal. El arreglo de medias *m* es una versión suavizada de la original. Podemos repetir exactamente el mismo proceso sobre esta nueva señal suavizada.

Podemos dar ahora una representación simbólica usando operadores. Obtendremos cuatro operadores, uno para cada paso del proceso. El operador L efectúa el suavizado m=(p+q)/2, resultando una versión suavizada de la mitad del tamaño. El operador H computa la amplitud de la wavelet w=(p-q)/2, dando los coeficientes wavelet de ese nivel. Para reconstruir la señal a partir de esos valores, el operador L* estira la señal de longitud n a 2n duplicando cada entrada. El operador H* toma cada amplitud wavelet w y la duplica como L*, pero niega el segundo valor. El operador L actúa como filtro pasabajos ; el operador L* es un simple operador de zoom. El operador H es un filtro pasaaltos ; H* también hace zoom pero niega el segundo valor.

3.5.1- Construyendo los operadores

Comenzamos definiendo el filtro pasabajos L. Las wavelets requieren un término de normalización, que puede ser $1/\sqrt{2}$ en ambas ecuaciones, o 1/2 sólo en una de ellas. Supongamos que tenemos una señal de entrada x de longitud 2^n. El filtro podría ser escrito como un operador $x^{(n)}$ que produce una nueva señal $x^{(n-1)}$ de longitud 2^{n-1}. Cada uno de los nuevos valores i está dado por la evaluación de Lx para un valor particular de i. L está construido por los coeficientes de la ecuación de dilación de la wavelet :

$$L_{ik} = C_{2i-k}$$

Usando esta fórmula podemos encontrar el resultado de aplicar L a $x^{(k)}$ para encontrar el valor i de $x^{(k-1)}$:

$$(Lx)(i) = \tfrac{1}{2} \Sigma_k C_{2i-k}\, x(k), \qquad i= 1..n/2$$

El operador L* resta la señal de baja resolución a una versión de mayor resolución directamente. Está definida similarmente a L, pero los coeficientes en la ecuación tienen las posiciones cambiadas.

$$(L^*x)(k) = ½ \Sigma_i C_{2i-k}\, x(k), \qquad k= 1..n/2$$

La señal diferencia de $d^{(n)}$ correspondiente al detalle perdido cuando la señal $x^{(n)}$ es suavizada obteniendo la señal $x^{(n)}$ está dada por :

$$d^{(n)} = x^{(n)} - L^* x^{(n-1)} = (I - L^*L)\, x^{(n)}, \qquad n= N..1$$

donde I es el operador identidad.

Ahora es fácil recuperar la señal original de la versión más suavizada y las señales diferencia $d^{(0)}$ hasta $d^{(N-1)}$.

$$x^{(n)} = d^{(n)} + L^* x^{(n-1)}, \qquad n= N..1$$

Para encontrar los coeficientes wavelets sólo necesitamos construir un filtro pasaaltos H, tal que HL*=I. Como L, este filtro también tiene ½ adelante. Los coeficientes de este filtro están dados por :

$$H_{ik} = (-1)^{k+1} C_{k+1-2i}$$

Ahora podemos dar el algoritmo completo para encontrar los coeficientes wavelet.

Descomposición : Dada una señal $x^{(N)}$ conteniendo 2^N , para n=N..1, computamos

$$x^{(n-1)} = L\, x^{(n)}$$

$$b^{(n-1)} = H\, x^{(n)}$$

Reconstrucción : Dados $x^{(0)}$ y $b^{(0)}$ ... $b^{(N-1)}$, para n=1..N, computamos

$$x^{(n)} = L^* x^{(n-1)} + H^* b^{(n-1)}$$

3.6- Notación de Matriz

Una manera compacta y fácil de manipular la notación para computar las transformaciones es usando matrices. Suponiendo que c_{ik}, i=0..F-1, son los coeficientes de la ecuación de dilación, entonces la matriz [L] está definida por $L_{ik} = ½\, c_{2i-k}$. La matriz [H] está definida por $H_{ik} = ½\, (-1)^{k+1} c_{k+1-2i}$.

Las matrices de reconstrucción en los casos ortogonales son las traspuestas de $[L^*]=[L]^T$ y $[H^*]=[H]^T$.

	c1	c0	0	0				
	0	0	...				0	0
	0	0	c1	c0				
[L]=				...	...			
	0	0	...			c1	c0	0
	0	0	...				0	0
	c0	0	...				0	c1

	c0	-c1	0	0				
	0	0	...				0	0
	0	0	c0	-c1				
[H]=				...	...			
	0	0	...			c0	-c1	0
	0	0	...				0	0
	-c1	0	...				0	c0

3.6.1- Condiciones de coeficientes

Podemos juntar las operaciones de matrices de la sección previa en un simple cálculo intercalado. Cuando tratemos de invertir esta matriz de ecuaciones encontraremos que requiere algunas condiciones sobre los coeficientes, lo cual a su vez influencia los tipos de funciones que pueden ser usadas para satisfacer la ecuación de dilación.

Notar que tanto L como H submuestrean sus entradas en un factor de 2. Por lo tanto la forma de la matriz de estos operadores tienen dimensiones n x 2n. Podemos ponerlas juntas, intercaladas, para crear una simple matriz A.

	c0	c1	0	0				
	0	0	-c1	c0				
[A]=					...			
	0	0			...	c0	c1	0
	c0	0			...	0	0	-c1

Para invertir la transformada, necesitamos encontrar $A^{-1}(A^T)$.

	c0	0	...		c0
	c1	0	...		0
	0	-c1			
$[A]^T$=	0	c0			
			...		
				c0	0
				c1	0
				0	-c1

Multiplicando estas dos encontramos :

	α	0	β	0	...	0	0	β	0
	0	α	0	β	...	0	0	0	β
	β	0	α	0	...	0	0	0	0
$[AA^T]$=	0	β	0	α	...	0	0	0	0
	...	...	...	...	...	...	...	...	...
	0	0	0	0	...	α	0	β	0
	0	0	0	0	...	0	α	0	β
	β	0	0	0	...	β	0	α	0
	0	β	0	0	...	0	β	0	α

donde, $\alpha = c0^2 + c1^2$
$\beta = c0*c2 + c1*c3$

El producto de AA^T será la identidad si $\alpha = 1$ y $\beta = 0$. Estas son las primeras dos condiciones que se demandarán a los coeficientes de la ecuación de dilación.

1) $c0^2 + c1^2 = 1$
2) $c0*c2 + c1*c3 = 0$

En el caso que c2 y c3 no existan, se consideran c2=c3=0.

Las wavelets de Haar son de orden cero, es decir, pueden hacer match sólo con funciones constantes a trozos. Se dice que el orden de la wavelet es el tipo de señal con la que pueden hacer match la función de escala. La regularidad de cada función es el número de veces que la función es diferenciable de manera continua. Podemos movernos del orden cero ofrecido por las wavelets Haar a una continuidad de primer orden imponiendo dos condiciones adicionales:

3) $c3 - c2 + c1 - c0 = 0$
4) $0c3 - 1c2 + 2c1 - 3c0 = 0$

Cuando estas dos condiciones se juntan con las otras dos, obtenemos cuatro coeficientes que generan wavelets que pueden hacer match con cualquier función lineal.

3.7- Análisis de Multiresolución

Cada aplicación del filtro pasabajos reduce a la mitad el número de muestras que representan nuestra señal. Podemos pensar en estas series de señales como representando la señal original en una variedad de resoluciones.

La técnica del análisis de multiresolución provee un formalismo para tratar esta propiedad de las wavelets. Podemos pensar en la señal de entrada como perteneciente a un espacio de señales, y entonces construir una cadena anidada de tales espacios, cada uno conteniendo la señal a una resolución menor.

La función de escala v(t) implica un espacio V_0, que es el espacio de todas las funciones cv(t) para alguna constante c. Similarmente, la primera wavelet $W^0(t)=w^{0,0}(t)$, implica un espacio W_0, que es el espacio de todas las funciones $cw^0(t)$. Estos dos espacios son ortogonales por construcción:

$$\int [av(t)][bw^0(t)] = 0$$

Si combinamos estos dos espacios por una suma cartesiana, generamos un tercer nuevo espacio V_1 $\qquad V_1 = V_0 \oplus W_0$

Este nuevo espacio contiene las funciones que son combinaciones de los dos subespacios:

$$V_1 : f(t) = av(t) + bw^0(t)$$

Éstas están ilustradas en la figura 15 para las wavelets de Haar. Por lo tanto v1 contiene todas las funciones construidas de dos copias de la función de escala original individualmente hechas escala, dilatadas y trasladadas.

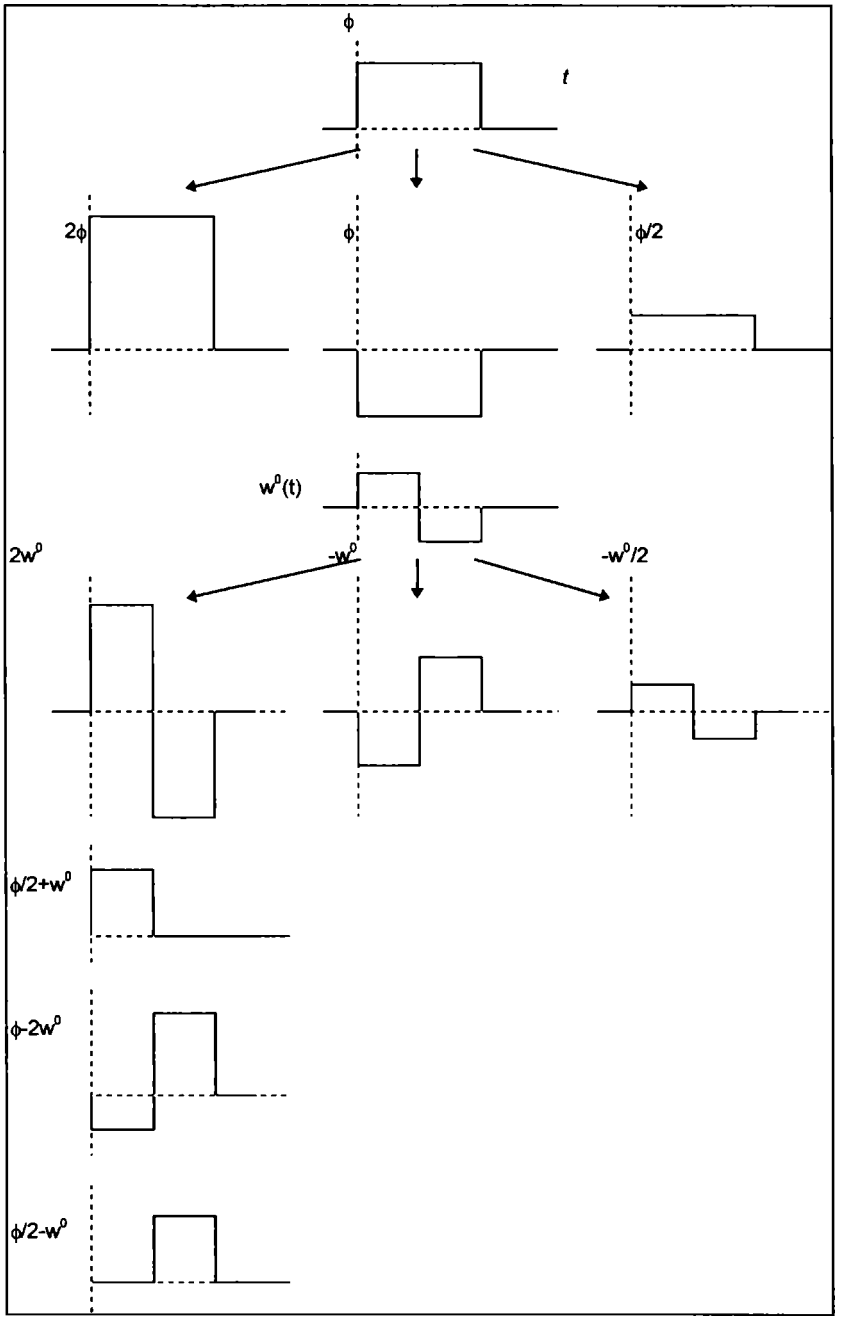


Figura 15

Hay dos puntos esenciales a notar sobre nuestro nuevo espacio de funciones V_1. El primero es que fue construido combinando un espacio con un segundo espacio ortogonal. Por ejemplo, considere el espacio P_0, que contiene todos los polinomios de orden 0; esto es, todas las funciones constantes $f_0(x)=c0$ para algún $c0 \in R$. Deberíamos tener otro espacio P_1 que contenga todos los polinomios de orden 1, o funciones lineales $f_1(x)=c1x+c0$ para $c1,c0 \in R$, y un espacio P_2 para cuadráticos $f_2(x)=c2x^2+c1x+c0$, y así. Podemos construir tantos espacios como queramos , donde cada P_k contiene los polinomios $f_k(x)=\Sigma^k_{i=0}cix^i$.

El segundo punto importante sobre esta construcción es que los espacios están anidados. Todas las funciones $f_k(x)$ en el espacio P_k están contenidas en el espacio mayor P_{k+n}, para todos los $n > 0$. Escribimos esta secuencia de espacios anidados como $P_0 \subset P_1 \subset P_2$. En términos de nuestra construcción de V_1, hallamos que $V_0 \subset V_1$, puesto que todas las funciones $av(t)$ están en el espacio de las funciones $av(t) + bw^0(t)$. En general, podemos construir una secuencia de espacios cerrados anidados V_i, tales que:

$$\ldots \subset -V_2 \subset -V_1 \subset V_0 \subset V_1 \subset V_2 \subset \ldots$$

Ahora estamos listos para describir el marco de trabajo de multiresolución para wavelets. Cada espacio V_m está hecho de todas las combinaciones lineales de las funciones $v(2^mt-k)$, y cada espacio W_m está hecho de todas las combinaciones lineales de las funciones $w^0(2^mt-k)$. En general construimos cada espacio V_{m+1} combinando el espacio V_m con un espacio ortogonal W_m recursivamente:

$$V_{m+1} = V_m \oplus W_m$$

$$= V_0 \oplus W_0 \oplus W_1 \ldots \oplus W_m$$

Resumiremos estas propiedades con respecto a todas las funciones $f \in L^2(R)$, es decir las que satisfacen $\int |f(t)|^2 < \infty$. En general, un análisis de multiresolución es un conjunto cerrado de subespacios V_m , $m \in Z$, con las siguientes propiedades:

1. Contención: $V_m \subset V_{m+1}$.
2. Distinción: $\cap_m V_m = 0$.
3. Completitud: $\cup_m V_m = L^2(R)$.

4. Desplazamiento: si f(t) $\in V_m$, entonces f(t-k) $\in V_m$ para todos los k $\in Z$.

5. Escala: si f(t) $\in V_m$, entonces f(2t) $\in V_{m+1}$ para todas las f $\in L(R)^2$.

6. Base: Existe una función de escala v(t) $\in V_0$, tal que para todo m, el conjunto de funciones

$$\{v_{mn}(t)=2^{-m/2}v(2^{-m}t-n)\}$$

forma una base ortonormal para V_m; esto es

$$\int v_{mp}(t)v_{mq}(t)dt = \begin{cases}1, p=q\\ 0, sino\end{cases}$$

3.8- Wavelets en el Dominio de Fourier

Es instructivo comparar la localización en tiempo-frecuencia de la transformada wavelet y la transformada Fourier short-time(por ventanas), como se ve en la Figura 16. En esta figura un punto representa el centro de cada transformada; la posición horizontal es el tiempo y la vertical es su frecuencia. Note que en la STFT, los puntos están regularmente espaciados en ambas direcciones. Se posiciona la ventana en el tiempo nt_0; valores enteros de n producen un espaciado horizontal uniforme. Se toma la transformada en la frecuencia mw_0, produciendo un espaciado vertical uniforme. Por lo tanto, una vez que se han elegido los valores de t_0 y w_0, se genera una familia completa de transformadas a incrementos uniformes. Esto sucede porque se tiene una ventana constante para todos los rangos de tiempo y frecuencia.

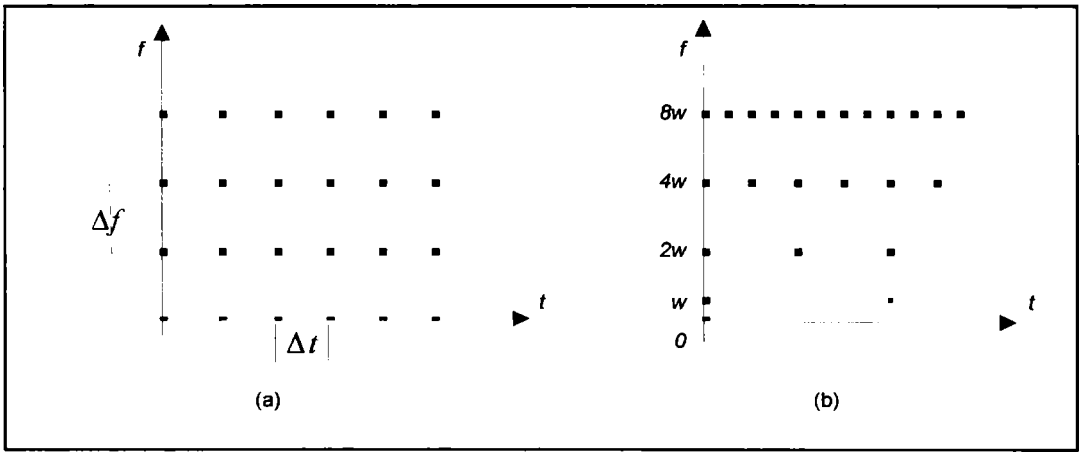


Figura 16 - (a) STFT ; (b) Wavelet.
t representa el tiempo, f la frecuencia.

La figura también contiene los patrones para la transformada wavelet. Nótese primero cómo el espaciado horizontal se ajusta al de la frecuencia; mientras la frecuencia crece y el soporte de la wavelet en dominio temporal decrece, las copias de la wavelet necesitan ser localizadas más cerca unas de otras para cubrir el dominio. También nótese que el espaciado en frecuencia crece geométricamente. Esto implica que el cociente entre el ancho de banda de la wavelet y el centro de su frecuencia es constante.

Se pueden comparar las transformadas wavelet y fourier short-time de otra manera también. Un método común para mostrar el contenido frecuencial de las señales de 1D es usar la STFT para computar un espectrograma. Ésta es una técnica común para mostrar el contenido espectral de señales que varían en el tiempo. El correspondiente diagrama para wavelets es llamado espectrograma de wavelet o escalograma. En las Figuras 17 se ve el escalograma y el espectrograma correspondientes a un conjunto de tres ondas sinusoidales en f_0, $2f_0$ y $4f_0$. Nótese que la resolución frecuencial en el espectrograma es una constante Δf resultante de una ventana fija, mientras que la respuesta del escalograma se agranda con el incremento de la escala.

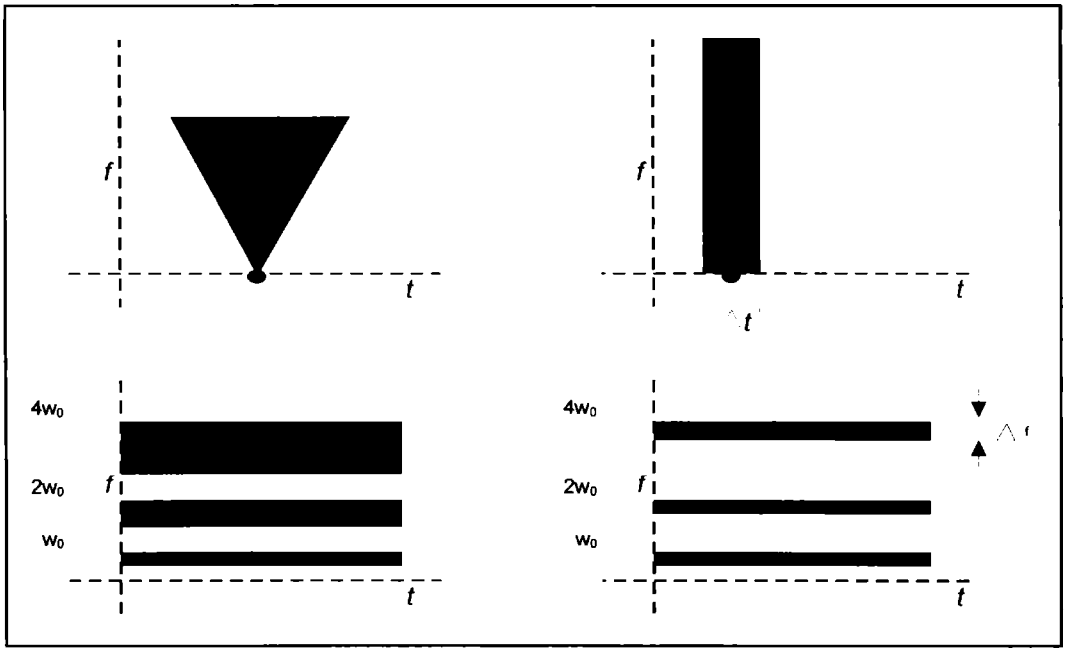


Figura 17 - Escalograma y espectograma de la función impulso.

Por lo tanto las wavelets se adaptan a la frecuencia a la cual son aplicadas. A bajas frecuencias, incluyen grandes secciones de la señal muy espaciadas. A mayores frecuencias, las wavelets son empaquetadas más juntas e incluyen trozos más pequeños de la señal. Ésta es la principal fuerza de las wavelets con respecto a la STFT.

3.9- Wavelets en dos Dimensiones

Tal como se construyen transformadas de Fourier para señales 2D, también podemos construir transformadas wavelet 2D. Se usan para almacenamiento y compresión de imágenes.

Existen varios métodos de construcción de wavelets multidimensionales. A continuación nos centraremos en los dos métodos más populares, llamados *rectangular* y *cuadrado* (también llamados *estándar* y *no estándar*).

Ambos métodos usan el *producto de tensor* de dos funciones *a(t)* y *b(t)*. Llamemos a las coordenadas 2D *x* e *y*. La idea es que a una de las funciones se le pase el parámetro *x*, y a la otra, el parámetro *y*, y el valor de la función 2D es el producto de las dos funciones ; por ejemplo, el valor en *(x,y)* es *a(x)b(y)*. Escribiremos esto como $(a \otimes b)(x,y)$.

3.9.1- Descomposición wavelet rectangular

La idea es hacer productos de tensores de todas las funciones involucradas en la transformada wavelet de 1D y usarlas en 2D.

En otras palabras, supongamos que tenemos un vector de entrada de cuatro elementos *x[n]*. Para hacer match con la base de Haar, tendremos los coeficientes de la función de escala v=(1,1,1,1), los cuatro elementos wavelet $w^{0,0}$=(1,1,-1,-1), y los dos elementos wavelets de dos $w^{1,0}$=(1,-1,0,0) y $w^{1,1}$=(0,0,1,-1). Para hacer el conjunto de bases 2D, construiremos 16 productos de tensores que se forman combinando esas cuatro funciones. Como cada función de entrada es de cuatro elementos de largo, la funciones base resultantes son matrices de 4x4.

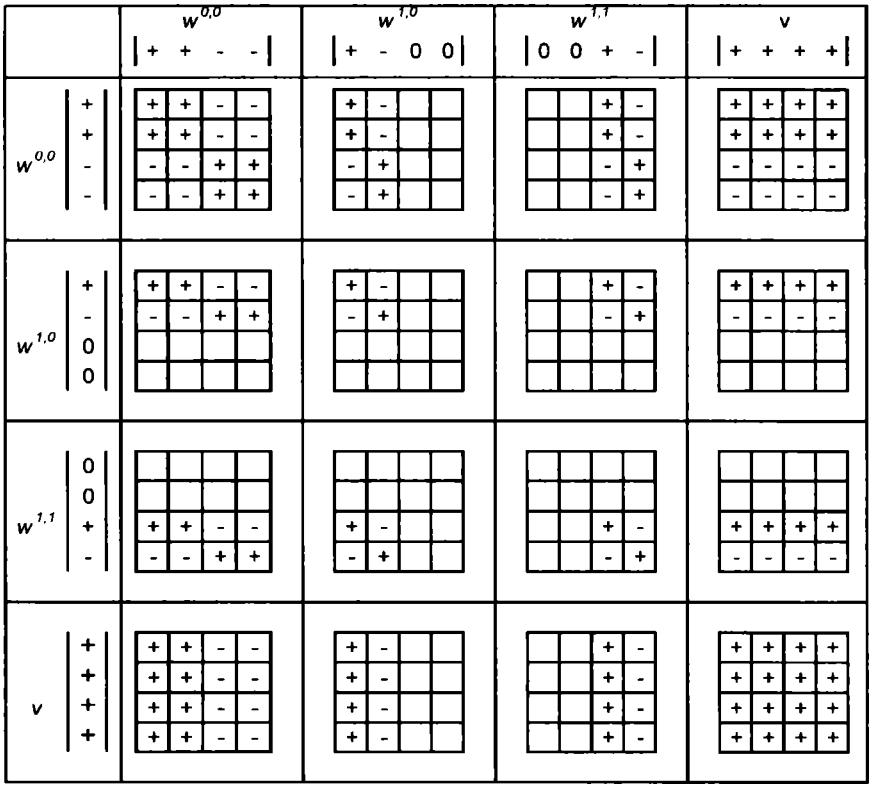


Figura 18 - Las 16 funciones base para una descomposición wavelet rectangular de una matriz de 4x4. Cada función está hecha por el producto de tensor de la función del tope de la fila con la función de la izquierda ; por ejemplo : la 3º función del tope de la fila está dada por $w^{1,1} \otimes w^{0,0}$.

Por lo visto en la figura, vemos que computamos la diferencia de información en varias direcciones. En particular, siguiendo la tabla, vemos que computamos información de alta frecuencia distribuida en tres direcciones : izq. a der. ($w^{1,0} \otimes v$), arriba a abajo ($v \otimes w^{1,0}$), y en diagonal ($w^{1,0} \otimes w^{1,1}$).

Para transformar una señal 2D en esas bases, por cada base multiplicamos la señal término a término, sumando el resultado, y dividiendo por un factor de normalización (en este caso igual al número de términos distintos de 0 de la función base). Notar que esto sólo funciona si las wavelets son duales, como es el caso de las bases de Haar.

3.9.2- Descomposición wavelet cuadrada

Hemos visto que las funciones bases construidas anteriormente tienen regiones cuadradas y rectangulares de elementos distintos de cero. Esto es útil cuando

queremos construir una descomposición con diferentes rangos de escala en dimensiones diferentes. Si queremos tratar todas las dimensiones de la misma manera, podemos construir una descomposición que tenga solo términos cuadrados ; esto es llamado descomposición *cuadrada* (o *no estándar*).

Una vez más construiremos las funciones bases de productos de tensores de la función de escala *v* y las wavelets $w^{a,b}$. Esta vez, sin embargo, nos guiaremos por una analogía con el análisis de multiresolución visto anteriormente.

Comenzamos con la función base más simple formada por $v \otimes v$; esto es, una caja. El espacio que comprende a todas las trasladadas de *v* contiene todas las funciones constantes sobre cuadrados de tamaño entero, y lo llamaremos $V_0^{(2)}$. Nos gustaría movernos a un espacio $V_1^{(2)}$, que fuera constante a trozos sobre cajas de la mitad del tamaño entero. Para moverse de un espacio de menor resolución a uno de mayor resolución, hacemos el producto cartesiano de $V_0^{(2)}$ con el espacio de detalle $W_0^{(2)}$, tal que

$$V_0^{(2)} \otimes W_0^{(2)} = V_1^{(2)}, \qquad W_0^{(2)} \perp V_0^{(2)}$$

Esto es, el espacio de detalle es ortogonal al espacio de baja resolución, y agrega justo a información que se necesita para obtener una mejor resolución. Eventualmente, $V_\infty^{(2)}$ será capaz de hacer match con todas las funciones 2D f(x,y).

Para hallar $W_0^{(2)}$, nuevamente tomamos una norma del caso de 1D y hacemos las cuatro funciones de productos de tensores que vienen de la combinación de la función de escala *v* y la primera wavelet $w^{0,0}$. Esto es, construimos $v \otimes v$, $v \otimes w^{0,0}$, $w^{0,0} \otimes v$, y $w^{0,0} \otimes w^{0,0}$. Estas cuatro funciones básicas se muestran en la figura 6.27, con los labels A, H, V y D respectivamente.

Estas cuatro funciones pueden hacer match con cualquier señal de 2 x 2. La descomposición puede ser escrita simplemente observando que

$$\begin{matrix} a & b \\ c & d \end{matrix} = g_A A + g_H H + g_V V + g_D D$$

Esto es, una matriz de entrada con elementos (a, b, c, d) es la suma de cuatro funciones básicas pesadas por sus coeficientes (g_A,g_H,g_V,g_D). Para encontrar estos coeficientes, podemos escribir la ecuación anterior como :

$$a= g_A+g_H+g_V+g_D$$

$$b= g_A-g_H+g_V-g_D$$

$$c= g_A+g_H-g_V-g_D$$

$$d= g_A-g_H-g_V+g_D$$

Ahora se formaron cuatro ecuaciones con cuatro incógnitas, de las cuales se pueden despejar los cuatro coeficientes.

3.10- Wavelets y Compresión de Señales

La idea subyacente de cualquier esquema de compresión es eliminar la correlación presente en los datos. Como se mencionó anteriormente algunos tipos de correlación son:

- correlación espacial (pixel),
- correlación frecuencial (componente frecuencial),
- correlación temporal (pixel de dos frames vecinos).

Cuando se conoce la correlación presente en los datos es posible encontrar la transformada óptima llamada Karhunen-Loève, sin embargo tiene varias desventajas:

- la matriz de correlación no se conoce,
- calcular los vectores de esa matriz tiene complejidad cúbica,
- aún conociendo la base óptima, calcular la transformada es un algoritmo cuadrático,
- la base depende del conjunto de datos.

Esto nos dice que necesitamos una transformada con las siguientes propiedades:

- independiente de los datos,
- que exista un algoritmo rápido para calcularla,
- que sea capaz de eliminar la correlación de un conjunto general de datos.

Una candidata posible es la FFT pero no cumple con la última propiedad.

Necesitamos una base local en espacio y en frecuencia que puede obtenerse:

- dividiendo el espacio frecuencial en ventanas y usando Fourier sobre cada una (base trigonométrica),
- usando una base Wavelet.

3.11- Medida del Error

Para los esquemas de compresión con pérdida usualmente se quiere que la imagen comprimida tenga la misma calidad visual que la original.

Se usan las siguientes medidas para comparar la imagen original (*fi*) y la comprimida (*fi'*).

- Raíz del error cuadrático medio:

$$\text{RMSE}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(fi-fi')^2}$$

- Pico del cociente señal-ruido (en dB):

$$\text{PSNR}=20\log_{10}\frac{max_i|fi|}{RMSE}$$

3.12- Compresión de Imágenes

Uno de los algoritmos más usados para la compresión de imágenes es el JPEG. El algoritmo divide la imagen en bloques de 8X8 pixels usando en cada uno la Transformada Discreta del Coseno (DCT). La desventaja de esto es que la imagen comprimida revela los bloques y no puede aprovecharse la correlación entre bloques.

Un algoritmo de compresión esencialmente consta de tres pasos: transformada, cuantificación, y codificación.

Transformada wavelet:

Para la elección de una determinada wavelet, deben considerarse las siguientes propiedades:

- Soporte compacto: los filtros deben ser FIR finitos.
- Coeficientes racionales: permiten evitar las operaciones de punto flotante.
- Suavidad: si la wavelet no es suave el error será fácil de detectar visualmente.

Longitud de los filtros: son preferibles los filtros cortos, pero hay un trade-off entre estos y la suavidad ya que la misma es proporcional a la longitud de los filtros.

Cuantificación

Un problema que impide la codificación eficiente es el hecho de que los coeficientes de la transformada pueden tener valores arbitrarios. El propósito de la cuantificación es restringir los valores de los coeficientes a un número limitado de posibilidades.

Codificación

El paso de codificación involucra reemplazar, de un modo reversible, el string de símbolos de entrada del cuantificador por un flujo de bits.

Las dos categorías principales son codificación de longitud fija y de longitud variable. En un codificador de longitud fija cada símbolo es reemplazado con el mismo número de bits. Es por lo tanto esencial usar un buen cuantificador. Un ejemplo es el algoritmo de Lloyd-Max.

Una variante más poderosa usa codificación de longitud variable. La idea es asignar las palabras más cortas a los símbolos más frecuentes. Supongamos que una palabra de código k_i tiene probabilidad p_i con

$$\sum_i p_i = 1$$

El contenido de información o entropía ahora está dado por

$$H = -\sum_i p_i \log_2 p_i$$

y esta es la cantidad mínima teórica de bits necesarios por palabra de código. El problema es que *H* no es necesariamente un número natural.

Los codificadores de longitud variable(o codificadores entrópicos) tratan de acercarse lo más posible a este mínimo. Los dos métodos más populares son Huffman y la codificación aritmética.

Debe tenerse en mente que estos codificadores son sólo óptimos es el caso en el que las probabilidades p_i son conocidas. En la práctica uno usualmente tiene que estimar p_i basado en los datos o en alguna información a priori.

Evidentemente, la posición de los coeficientes que fueron seteados a cero tiene que ser codificada también. Esto puede ser hecho con codificación Run Length, la cual es usualmente seguida de codificación entrópica de las longitudes de las corridas.

4- CUANTIFICACIÓN ESCALAR

4.1- Introducción

En su forma más simple, un cuantificador toma un número y selecciona el valor aproximado más cercano de un conjunto finito predeterminado de valores numéricos permitidos. Más precisamente, se define un cuantificador escalar *Q* como el mapeo *Q*:*R*—>C donde *R* es la línea real y

$$C \equiv \{y_1, y_2, \ldots, y_N\} \subset R$$

es el conjunto de salida o codebook con tamaño |C|=N. Los valores de salida,y_i, son llamados niveles de salida, puntos de salida o valores de reproducción. Se define la resolución o tasa de codificación, r, de un cuantificador escalar como r=log2N el cual mide el número de bits necesarios para especificar de manera única el valor cuantificado. La resolución indica la exactitud con la que el valor original es descripto. Asociada con cada uno de los N puntos de un cuantificador existe una partición de la línea real R en celdas o átomos r_i, i=1,2,...,N. La i-ésima celda está dada por $Ri=\{x \in R: Q(x)=yi\} \equiv Q\text{-}1(yi)$, la imagen inversa de y_i bajo Q. Se deduce de esto que $\cup Ri=R$ y $Ri \cap Rj=\varnothing$ para $i \neq j$. Una celda no acotada es llamada celda sobrecargada y cada celda acotada es llamada celda granular. Todas las celdas sobrecargadas forman una región sobrecargada, y todas las celdas granulares forman una región granular. Un cuantificador se dice regular si

a) cada celda R_i es un intervalo de las forma (x_{i-1},x_i) junto con una o más de las cotas, y

b) $yi \in (x_{i-1},x_i)$.

Los valores xi son a menudo llamados puntos cota, puntos de decisión, niveles de decisión o puntos finales. Un cuantificador regular tiene la propiedad de que si dos puntos a y b con a<b son cuantificados con el mismo valor w, cualquier otro valor entre a y b también será cuantificado con la misma salida w.

4.2- Medidas de la Performance de un Cuantificador

El propósito de la cuantificación es proveer una descripción de precisión limitada de un valor previamente desconocido. Por ello la entrada debe ser modelada como una variable random. Consecuentemente, el error introducido al cuantificar este valor es

también random. La medida más común de distorsión entre dos números es el error cuadrático definido por:

$$d(x,x')=|x-x'|^2.$$

Otra medida de interés es el error absoluto$|x-x'|$. Ambos son casos especiales de la m-ésima potencia de la magnitud del error:

$$d_m(x,x')=|x-x'|^m.$$

Para una medición del error a través del tiempo de vida del cuantificador el error absoluto del peor de los casos(entrada acotada), llamado máximo error, es el máximo de $d_1(x,x')$ tomado sobre todas las posibles entradas x. El promedio estadístico de la distorsión es a menudo una medida de performance más significativa e informativa; en general puede escribirse como

$$D=Ed(X,Q(X))=\int_{-\infty}^{\infty} d(x,Q(x))f_X(x)dx.$$

El promedio estadístico del error cuadrático(ECM) es referido como distorsión promedio.

Dada una señal $\{X_n\}$ y un cuantificador escalar Q, el error inevitable $e_n \equiv Q(X)-X_n$ que surge en la cuantificación es a menudo visto como ruido introducido por el cuantificador. Específicamente, el *ruido granular* es aquel componente del error de cuantificación que se debe al carácter granular del cuantificador para una entrada que cae dentro de las celdas acotadas del cuantificador. El *ruido de sobrecarga* es el error de cuantificación que es introducido cuando la entrada cae en una celda no acotada. Generalmente el ruido granular es relativamente pequeño en amplitud y ocurre en una variedad de grados, mientras que el ruido de sobrecarga es muy grande en amplitud pero ocurrirá muy rara vez para un cuantificador bien diseñado.

La performance de un cuantificador es a menudo especificada en términos del cociente señal-ruido (SNR), definido como la normalización de la energía de la señal dividido la energía del error de cuantificación y tomando un logaritmo a escala:

$$\text{SNR} = 10 \log_{10} \frac{E(X^2)}{D}$$

medido en unidades de decibeles(dB). Puesto que el tamaño del error de cuantificación es más significativo relativo a la energía de la señal, el SNR es una muy importante medida de performance orientada al usuario.

La secuencia de errores proveniente de aplicar un cuantificador ha sido definida como

$$e_n=Q(x_n)-X_n.$$

con lo cual $Q(X_n)=X_n+e_n$;

Esta representación es a menudo llamada *modelo de ruido aditivo* de un cuantificador y permite ver la cuantificación como la suma de un término ruido a la señal original.

4.3- Cuantificación Uniforme

El más común de los cuantificadores escalares es el cuantificador uniforme, a veces llamado lineal. La familia de operaciones de truncado y redondeo de números reales son ejemplos de cuantificación uniforme. Un cuantificador uniforme es un cuantificador regular en el cual (a) los puntos frontera están igualmente espaciados y (b) los niveles de salida para celdas granulares son los puntos medios del intervalo de cuantificación. El error máximo posible $\Delta/2$ es decir la mitad de la longitud cada celda de cuantificación. Para aislar el efecto del ruido granular sin que la señal sea acotada, se define el cuantificador reticulado como un cuantificador uniforme con un número infinito de niveles donde la recta real es particionada en un conjunto(contable) de intervalos de igual longitud Δ.

4.4- Cuantificación no Uniforme y Companding

Hay dos ventajas principales en el uso de niveles no uniformemente espaciados de cuantificación. Primero, es posible incrementar significantemente el rango dinámico que puede ser acomodado para un número dado de bits de resolución usando un cuantificador no uniforme adecuadamente elegido. Segundo, es posible diseñar un cuantificador hecho a medida de las estadísticas de una entrada específica con lo cual se obtiene un SNR mayor para una resolución dada y una fdp de entrada dada.

Un modelo general para cualquier cuantificador no uniforme con un número finito de niveles es aquel que transforma la entrada x con una función monotónica no lineal, para producir una salida $y=G(x)$, que entonces es cuantificada con un cuantificador uniforme produciendo y' y finalmente es transformada con la inversa G^{-1}. La salida final cuantificada es $x'=G^{-1}(y)$. La primer función no lineal G es llamada compresor y la inversa G^{-1} es llamada expansor. La estructura completa de un cuantificador no

uniforme consistente en un compresor un cuantificador uniforme y un expansor es llamada modelo de cuantificador *compandor*. La familia más importante de compandors es la logarítmica.

Aunque las características de diferenciabilidad y suavidad del compresor son convenientes para manipulaciones matemáticas, hay problemas para implementar con precisión las funciones no lineales. La tecnología de hoy permite la implementación precisa de cuantificadores uniformes o de características de un compresor lineal a trozos que puedan aproximar la suavidad de las curvas del compresor logarítmico u otros. Un *cuantificador uniforme a trozos* es un cuantificador cuyo rango consiste en varios segmentos, cada uno de los cuales contiene varias celdas de cuantificación y puntos de salida correspondientes a un cuantificador uniforme. Dentro de cada segmento el cuantificador resulta ser un cuantificador uniforme con un tamaño de celda particular; sin embargo, diferentes segmentos pueden tener diferentes tamaños de celda.

4.5- Condiciones para Optimizar

El objetivo principal de un diseño de cuantificador es seleccionar los niveles de reproducción y las regiones de partición o celdas tales que provean la mínima distorsión promedio posible para un número fijo de niveles N o, equivalentemente, una resolución fija r. De la descomposición estructural de un cuantificador, vimos que consta de un codificador y un decodificador. basados en ello se establecen dos condiciones críticamente importantes que son necesarias para la optimización. Específicamente, la parte de codificador de un cuantificador óptimo debe ser óptima para un decodificador dado, mientras que el decodificador debe ser óptimo para el codificador dado.

El mejor codificador para un *codebook* dado es el que satisface la condición del vecino más cercano, que requiere que la i-ésima región de la partición consista de todos los valores más cercanos a y_i que a cualquier otro nivel de salida.

La segunda condición de optimización es alcanzada fijando un codificador(partición) y optimizando el decodificador (codebook). La condición centroide es tanto necesaria como suficiente para esta optimización si se usa la medida de distorsión del error cuadrático medio. La condición del centroide es simplemente la condición que el

nivel de salida óptimo, y_i, para la i-ésima celda de la partición es el centroide o centro de masa, de aquella parte de la fdp de la entrada que cae en la región R_i.

4.6- Algoritmo de Diseño de Cuantificador Lloyd

Algoritmo de Lloyd
Paso 1. Comenzar con un codebook inicial C_1. Setear m=1.
Paso 2. Dado un codebook, C_m, realizar la iteración de Lloyd para generar un codebook mejorado C_{m+1}.
Paso 3. Computar la distorsión promedio para Cm+1. Si ha cambiado en una pequeña cantidad con respecto a la última iteración, parar. Si no, setear m=m+1 e ir al Paso 2.

Iteración de Lloyd para mejorar el codebook:

(a)Dado un codebook, $C_m=\{y_i\}$, encontrar la óptima partición en celdas de cuantificación, esto es, usar la condición del vecino más cercano para formar las celdas del vecino más cercano:

$$R_i=\{x:d(x,y_i)<=d(x,y_i);\ \text{para todo } j\neq i\}.$$

(b)Usando la condición del centroide, hallar C_{m+1}, el codebook óptimo para las celdas recién encontrado.

Algoritmo Lloyd II
Paso1. Encontrar un valor inicial de $\{x_j; j=1,2,...,N-1\}$ y $\{y_j; j=1,2,...,N\}$. Setear $k=1, x_0=-\infty$.
Paso 2. Encontrar y_k tal que sea el centroide del intervalo (x_{k-1},x_k).
Paso 3. Hallar x_k tal que sea el punto medio de (y_k,y_{k+1}).
Paso 4. Si $k=N-1$ ir al Paso 5, si no setear $k=k+1$ e ir al Paso 2.
Paso 5. Computar c, el centroide del intervalo $(x_{N-1},+\infty)$. Si $\|y_N-c\| < \varepsilon$, parar. Si no, ir al Paso 6.
Paso 6. Sea $y_N = y_N - \alpha(y_N - c)$ y sea $k=1$. Ir al Paso 2.

5- CODIFICACIÓN ENTRÓPICA

5.1- Run-Length

La probabilidad p (n) de una corrida de exactamente n ceros(puesto que cada corrida de 0s termina con un 1)

$$p(n) = p^n q \quad (n = 0,1,...)$$

Sumando sobre todas las posibles longitudes n, tenemos

$$\sum_{n=0}^{\infty} p^n q = q \sum_{n=0}^{\infty} p^n = \frac{q}{1-p} = \frac{q}{q} = 1$$

como debía ser.

Supongamos por un momento decidimos enviar un número de k dígitos binario para representar una longitud de corrida. Por una razón veremos que podemos hacer esto solo para corridas de 0 a 2^k-2. El receptor copia el número indicado de 0s y seguido copia un 1. Si el número enviado es un 0, simplemente se copia un 1. Para el número binario (2^k-1) el receptor copia (2^k-1) 0s y no copia un 1. Si el siguiente número enviado es un 0, entonces por supuesto un 1 es copiado, y para cualquier otro número el número de 0s es copiado seguido de un 1. Luego, podemos enviar cualquier longitud de corridas de ceros que se necesite enviando una cantidad suficiente de números binarios de k dígitos, k>1.

5.2- Codificación Huffman

La ventaja de un código en el cual la longitud de los símbolos es variable es que a veces el código es más eficiente en el sentido que para representar la misma información usa menos dígitos en promedio. Sin embargo los códigos de longitud variable traen aparejado un problema fundamental: ¿En el receptor, cómo reconocen cada símbolo del código? La propiedad que se necesita es la decodificabilidad única.

Definición. La n-ésima extensión de un código es simplemente todas las posibles combinaciones de n símbolos del código fuente original. Hay q^n símbolos en la n-ésima extensión. Para decodificabilidad única, ningún para de concatenaciones codificadas pueden ser iguales, aún para diferentes extensiones.

5.2.1- Códigos instantáneos

En un código instantáneo cada bit del flujo recibido es examinado sólo una vez, y cuando un símbolo completo es recibido, el receptor lo reconoce inmediatamente, y no tiene que seguir esperando antes de decidir qué símbolo del mensaje se recibió. Ningún símbolo codificado de este código es un prefijo de otro símbolo.

Es claramente una condición tanto necesaria como suficiente que un código instantáneo no tiene ninguna palabra de código s_i que sea un prefijo de otra palabra de código s_j.

5.2.2- Códigos Huffman

Se quiere que los símbolos más frecuentes tengan la codificación más corta. Si la probabilidad del i-ésimo símbolo es p_i y su longitud es l_i, entonces la longitud promedio es

$$L_{prom} = \sum_{i=1}^{q} p_i l_i$$

Sin pérdida de generalidad los p_i pueden ser tomado en orden decreciente.

$$p_1 >= p_2 >= p_3 >= \ldots >= p_q$$

y

$$l_1 <= l_2 <= l_3 <= \ldots <= l_q$$

entonces el código no es eficiente en el sentido que podría tenerse una longitud promedio menor resignando las representaciones de los códigos de los símbolos s_1, s_2, s_3, ...,s_q.

La primer cosa a probar es que los dos símbolos menos frecuentes de un código eficiente tienen la misma longitud de codificación. Supongamos que el código de máxima codificación tiene longitud l. Si hay sólo uno de tal longitud, entonces como el código es instantáneo, cualquier código de longitud l-1(o menor) no es prefijo del código de máxima longitud. Por lo tanto, la última parte del símbolo más largo podría ser eliminada sin pérdida de información en la decodificación. Luego los dos símbolos más largos deben tener la misma longitud, y deben ser los dos menos probables.

El método de codificación es una reducción en cada paso a un código más corto. Simplemente se combinan los dos códigos menos probables del alfabeto fuente en un único símbolo cuya probabilidad es igual a la suma de las dos probabilidades correspondientes. Luego, hay que codificar el alfabeto fuente con un símbolo menos.

Repitiendo esto paso a paso, se reduce al problema de codificar sólo dos símbolos del alfabeto fuente, lo cual es fácil, se usan el 1 y el 0. Ahora yendo hacia atrás, uno de estos dos símbolos es particionado en dos símbolos, y esto puede ser hecho agregando un segundo dígito 0 para uno de ellos y 1 para el otro. En el siguiente paso de regresión, uno de estos tres símbolos es particionado en dos símbolos del mismo modo. Y sigue así.

El proceso de codificación no es único en varios aspectos. Primero, la asignación de los símbolos 0 y 1 a lo dos símbolos de la fuente en cada etapa de partición es arbitraria, pero esto produce sólo diferencias triviales. Segundo, cuando dos probabilidades son iguales, pareciera ser un problema de indiferencia cuál se pone arriba en la tabla, aunque los códigos resultantes pueden tener distinta longitud de palabra. Sin embargo, en ambos casos la longitud promedio de la codificación de los mensajes en estos códigos será la misma. Los códigos tienen la misma eficiencia (longitud promedio) pero no la misma longitud de los símbolos.

¿Cuál debería ser elegido? Una elección muy razonable es aquél cuya longitud promedio varíe menos sobre el conjunto de mensajes. Por lo tanto se computa la varianza, y resulta con menor variabilidad el que mueve los símbolos producto de una unión lo más arriba posible.

CAPÍTULO III

DESCRIPCIÓN DEL ALGORITMO

1- INTRODUCCIÓN

A continuación describiremos el algoritmo que hemos desarrollado. En primer lugar comenzamos presentando la estructura de datos utilizada, luego detallamos las fases de la aplicación.

2- ESTRUCTURAS DE DATOS

Los datos del árbol de descomposición wavelet no son guardados en una estructura que refleje su forma pues haría más lento el proceso y aumentaría el uso de memoria. Por ello no se usan más que grandes buffers y los archivos definitivos en los cuales se guarda la información.

En el archivo de descomposición los nodos del árbol van guardándose en orden breath-first-search, es decir, por niveles. Por otro lado, los nodos intermedios que son usados en descomposiciones recursivas son mantenidos en archivos temporarios.

El archivo comprimido final contiene una cabecera con datos inherentes a los parámetros y especificaciones de las distintas fases de la compresión. La misma tiene el siguiente formato:

FILTRO (0 Haar, 1 Daubechies)	**RUN-LENGTH** (0 sin, 1 con)	**HUFFMAN** (0 sin, 1 con)	**LONGITUD IMAGEN**
1 bit	1 bit	1 bit	5 bits + 1 byte

+

MEDIA1 .. MEDIAn Media de cada nodo. (float)	MAXVAL1 .. MAXVALn Máximo valor de cada nodo. (float)	CANTBIT1 .. CANTBITn Cantidad de bits asignados a cada nodo. (byte)

n : nodos del árbol de descomposición.

Los buffers son fijos y son manipulados como matrices de dos dimensiones. Del mismo modo se manejan los coeficientes de los filtros, en arreglos dinámicos que simulan matrices cuadradas.

3- FASES DE LA APLICACIÓN

El algoritmo se divide en dos grandes fases : compresión y descompresión.

3.1- Compresión

1)- Ingreso de datos

Para la ejecución se requiere la entrada de un archivo de imagen (RAW) cuadrada cuya longitud de lado sea potencia de 2, y los siguientes datos a elegir:

⇒ Tipo de filtro a usar : Haar o Daubechies.

⇒ un grado de compresión (cuantificación) que varía entre mínimo, medio y máximo.

⇒ un umbral (cuantificación) que varía entre mínimo, medio y máximo.

⇒ opción de usar o no codificación Run-Length.

⇒ opción de usar o no codificación Huffman.

2)- Transformación wavelet

Comprende el pasaje de la imagen del dominio espacial al frecuencial a través de la aplicación sucesiva de filtros pasabajos y pasaaltos obteniendo los coeficientes wavelet que la representan. Involucra los siguientes pasos:

A- Construcción de filtros.

A partir de la longitud y coeficientes ingresados se generan los cuatro filtros QMF 2-D, LL(pasabajos en ambas direcciones), LH(pasabajos horizontal y pasaaltos vertical), HL(pasaaltos horizontal y pasabajos vertical), HH(pasaaltos en ambas direcciones).

Estos filtros se arman a partir de los filtros pasabajos (L) y pasaalto(H) de 1-D, los cuales a su vez se arman a partir de c0..cn-1 de la siguiente forma:

El filtro L es un vector de longitud n compuesto por cn-1..c0 en ese orden; los coeficientes del filtro H están dados por $ci*(-1)^{i+1}$, i:0..n-1.

Cada elemento de la matriz del filtro 2-D, se calcula de la siguiente forma:

LL(i,j)=L(i)*L(j)

LH(i,j)=L(i)*H(j)

HL(i,j)=H(i)*L(j)

HH(i,j)=H(i)*H(j), donde i=0..n-1, j=0..n-1.

Ejemplo: filtros de Haar

- coeficientes: 1,1 , longitud del filtro :2.

L

1	1

H

1	-1

LL

1	1
1	1

LH

1	-1
1	-1

HL

1	1
-1	-1

HH

1	-1
-1	1

B- Lectura:

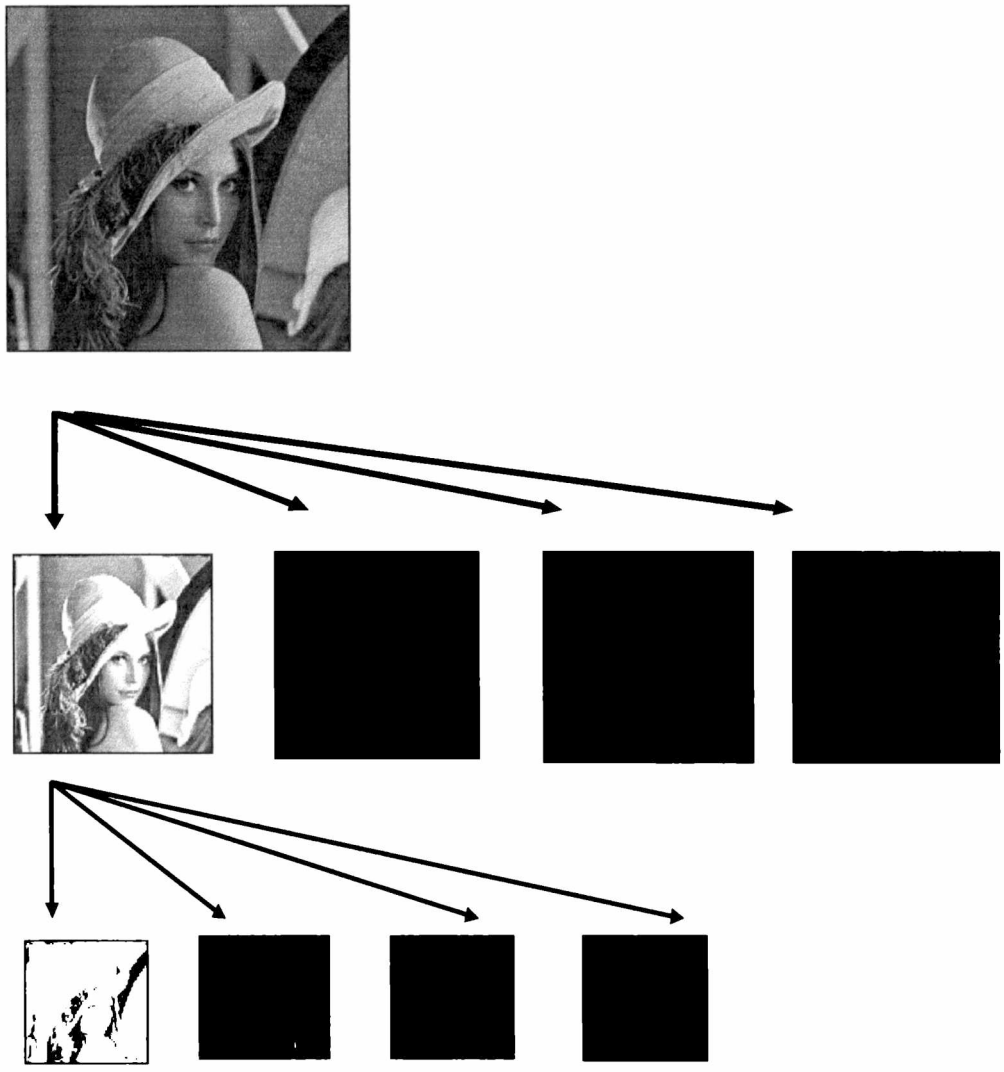
La lectura de la imagen se efectúa por ventanas rectangulares cuyo ancho es el de la señal y el largo es MaxRead/ancho.

Como el filtrado es cíclico, la primera vez se leen las últimas filas de la matriz seguidas de las primeras. Las siguientes veces se leen las últimas filas de la ventana leída anteriormente y se completa con las subsiguientes filas de la matriz.

C- Descomposición:

El proceso de naturaleza recursiva, se realiza por niveles a partir de la imagen original. Inicialmente se aplican los cuatro filtros sobre la matriz de la imagen completa, obteniéndose cuatro matrices que componen el nivel siguiente del

proceso, cuyo tamaño se reduce en cuatro por la decimación. En los sucesivos niveles la aplicación del filtro se efectúa sobre la matriz generada por el filtro LL.

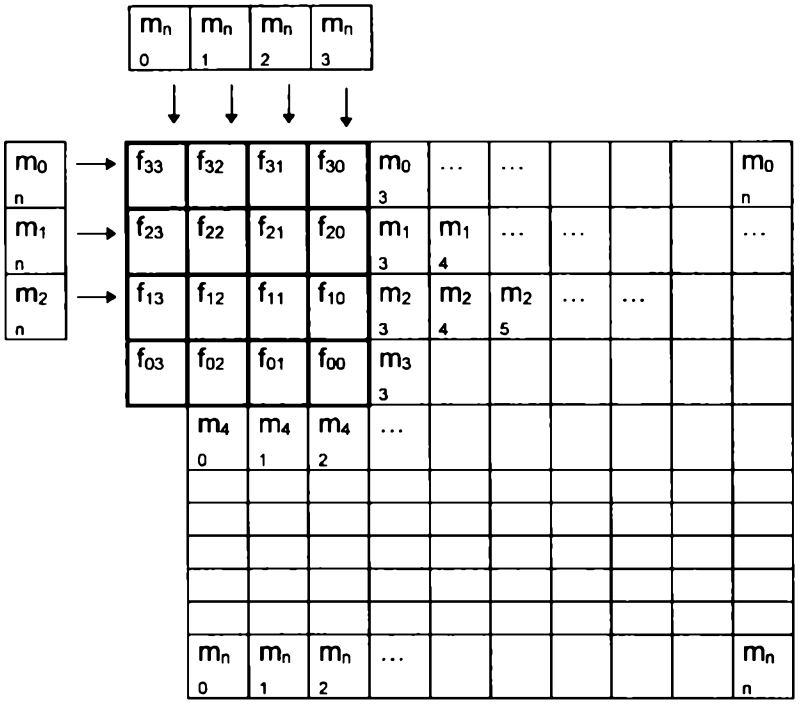


C.1 - Filtrado de una matriz

Consiste en aplicar el filtro apoyándolo sobre elementos intercalados de la matriz cada vez. De esta forma se produce la decimación tanto en las filas como en las columnas de la matriz de la imagen filtrada. Como la decimación es por dos en cada dimensión, la decimación total es por cuatro, es decir la matriz filtrada tendrá la cuarta parte del tamaño de la matriz de la cual proviene.

Para no agregar ceros (equivalentes a negro pues es cero luminosidad) alrededor de la imagen para el caso de filtrar los bordes, se efectúa un

filtrado circular. Tal forma de filtrado consiste en completar los datos faltantes "debajo" del filtro, con los datos del otro extremo de la fila o columna de la matriz de la imagen. De este modo cada fila o columna de la matriz funciona como un arreglo circular sobre el cual se aplican los filtros.



C.2- Aplicación del filtro

Dada F la matriz del filtro a aplicar y M la submatriz de la imagen sobre la cual se aplica el filtro, cada elemento de la matriz filtrada se calcula:

- sum=0
- for i=0..n
- for j=0..n
- sum=sum + M(n-1-i,n-1-j) * F(i,,j)

(depende de n, longitud del filtro).

Como puede leerse en el seudocódigo, el producto de convolución entre la submatriz y el filtro, se realiza hacia adelante en el filtro, pero hacia atrás en la matriz, de manera que se comienza a filtrar sobre el extremo inferior derecho de la submatriz de imagen, con el extremo superior izquierdo del filtro.

Ejemplo:

1	-1
-1	1

filtro

3	2	4	6
4	3	6	2
4	1	5	3
2	3	1	5

imagen

0	4
-1	1

resultado

3)- Cuantificación

Con el árbol de descomposición armado se procede a reducir el tamaño de la información con una pérdida irreversible. Éste es el paso que realiza la compresión real de la señal.

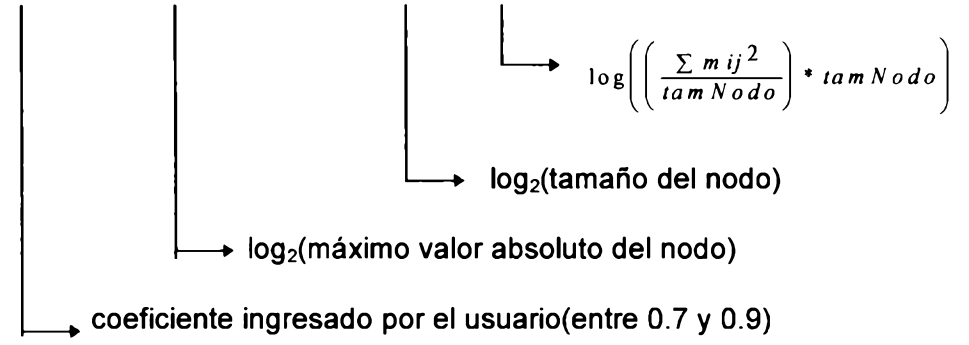
A- Normalización

Sobre los datos del árbol de descomposición wavelet se realiza una normalización que consiste en restarle a los coeficientes de cada nodo la media del mismo, obteniendo así una distribución de probabilidades con media 0.

B- Cálculo de coeficientes para la asignación de bits

Primero se busca el coeficiente de mayor valor absoluto para cada nodo, pues es el factor principal en la asignación de los bits. Simultáneamente se calcula la proporción de energía que aporta cada nodo al total. Se considera como energía a la suma del cuadrado de los valores de cada nodo y la proporción es tomada como el cociente entre la energía de un nodo y la energía total de la imagen transformada. Se toma en cuenta además un peso de cada nodo, que es el logaritmo de un valor inversamente proporcional al tamaño del lado de cada uno.

bits = coef * min(maxrep, maxrep * (peso/energia))



C- Asignación de bits

Todos los coeficientes calculados en el paso anterior, junto con un nivel de compresión ingresado por el usuario(bajo, mediano o alto), es utilizado para el cálculo de los bits que le serán alocados a cada nodo.

D- Establecimiento del Umbral y cuantificación

Una vez asignados los bits, se recorre el árbol seteando a 0 los coeficientes cuyo módulo es menor que un umbral dado y luego efectivizando la cuantificación. Esto último se logra dividiendo los coeficientes de cada nodo por un valor que reduce su magnitud tal que el máximo coeficiente pueda representarse con la cantidad de bits asignados.

- dato cuantificado = redondear(abs(valor)/máximo valor absoluto del nodo)
- ponerNBits(dato cuantificado,bits asignados al nodo)

E- Almacenamiento

Sólo los bits asignados a cada nodo son guardados para cada valor cuantificado. Juntamente con esta información, se almacena la cantidad de bits asignados a cada nodo, para la posterior decuantificación.

4)- Codificación

Una vez cuantificados los coeficientes se codifican entropicamente para suprimir la redundancia existente en el lenguaje binario y acercar la longitud promedio lo más posible a la entropía de los datos.

Esto se logra con codificación Run-Length y luego codificación Huffman.

A- Codificación Run Length:

Para suprimir los ceros consecutivos que aparecen naturalmente después de cuantificar, por el establecimiento del umbral y la reducción de precisión, se utiliza el código Run Length, que es tanto mejor cuanto más largas sean las corridas de ceros. Se da la posibilidad al usuario de utilizar o no estos códigos para comprobar su efectividad y aporte a la compresión total.

B- Codificación Huffman:

Con el objeto de acercar la longitud promedio de los datos al límite inferior que es la entropía, se aplica el código Huffman. Se da la posibilidad al usuario de elegir entre Huffman estático o dinámico, pudiendo además en el primer caso proveer una tabla o aceptar la que está por defecto. En el caso de codificación dinámica hay un paso anterior que no se efectúa en la codificación estática:

B.1- Armado de la tabla Huffman

Para armar la tabla de codificación dinámica deben hacerse estadísticas sobre los datos a codificar con lo cual los mismos son recorridos dos veces, una al armar la tabla y otra al reemplazar los datos por sus respectivos códigos.

a) *Estructuras de datos :*

La tabla se manipula a través de una lista implementada con un arreglo, cuyos componentes contienen un símbolo, su probabilidad, la dirección del elemento anterior de la lista y del siguiente (ordenados de mayor a menor por probabilidad), la dirección de los dos sumandos que componen la probabilidad (si es resultado de una suma).

De este modo se facilita la ordenación de las probabilidades iniciales(una por cada símbolo), y la inserción de las nuevas probabilidades, que son resultado de la suma de las dos menores.

b) Extracción de estadísticas :

En esta parte del proceso se recorre el archivo de datos contando la cantidad de veces que aparece cada símbolo y luego dividiendo esa cantidad por el total de símbolos contenidos en el archivo, lo cual constituye la probabilidad de aparición de cada símbolo en ese conjunto de datos.

c) Formación de los códigos :

Una vez calculadas las probabilidades de los símbolos, las mismas son ordenadas de mayor a menor. Entonces se van agrupando las dos últimas probabilidades para formar una nueva, igual a la suma de ambas, sucesivamente hasta llegar a tener sólo dos. Durante este procedimiento, se van haciendo los enganches necesarios en la lista que manipula la tabla Huffman para el siguiente paso.

Seguidamente se vuelve hacia atrás restaurando los sumandos de cada nueva probabilidad, y al mismo tiempo asignando un nuevo bit de código a cada par de probabilidades mayores, hasta tener nuevamente la lista de probabilidades de los símbolos originales, con sus respectivos códigos.

Una vez completo el proceso de asignación de códigos, se vuelca el contenido de la lista que manipula la tabla en un arreglo unidimensional de longitud igual a la cantidad de símbolos codificados, donde cada posición guarda el código igual a su índice.

B.2- Asignación de códigos

Con los códigos contenidos en la tabla de Huffman, se recorre el archivo reemplazando cada símbolo por su correspondiente código Huffman.

3.2- Descompresión

1)- Decodificación

Se aplica el decodificador Huffman, si así corresponde, y luego el descompresor Run Length.

A- Decodificador Huffman

Haciendo uso de la tabla Huffman, almacenada en archivo aparte si es estático, o en el archivo comprimido si es dinámico, se restauran los valores originales a partir de los códigos Huffman.

Se obtiene un bit por vez y se busca en la tabla si la concatenación de bits actual corresponde a un código Huffman. Si es así, se reemplaza el código por el símbolo respectivo, si no, se obtiene otro bit y se repite el proceso.

B- Descompresor Run Length

Se descomprime el archivo volviendo a poner en su lugar los ceros suprimidos. Esto se logra escribiendo la cantidad indicada de ceros cada vez que se encuentra una marca de Run Length seguida del número correpondiente de ceros codificados.

2)- Decuantificación

Se revierte en parte el proceso de cuantificación multiplicando los coeficientes de cada nodo por el valor correspondiente, volviendo así a la representación floating point. La pérdida se produce por el establecimiento del umbral, que no tiene proceso inverso, y por la reducción de precisión en la representación de los coeficientes.

- dato cuantificado = obtenerNBits(bits asignados al nodo)
- valor = dato cuantificado * máximo valor absoluto del nodo

A- Desnormalización:

Para completar la restauración de los datos wavelet se le suma a cada coeficiente la media de su nodo, para que éste recupere la distribución de probabilidades original.

3) -Antitransformación wavelet

La imagen, que para ser comprimida había sido transformada pasando del dominio espacial al frecuencial, ahora debe ser antitransformada para retornar al dominio original. Para ello se aplica la transformada inversa wavelet, que consiste en aplicar los filtros inversos , tanto pasabajos como pasaaltos, a la imagen transformada desde las hojas del árbol de descomposición(último nivel), hasta la raíz. El proceso involucra los siguientes pasos:

A- Construcción de filtros inversos:

A partir de la longitud y coeficientes ingresados se generan los cuatro filtros QMF 2-D, LL'-LH'-HL'-HH' de manera análoga a los filtros usados para la descomposición.

Los filtros inversos de una dimensión son los transpuestos de L y H. Esto significa que $L_i^{-1}=L_{n-1-i}$ y $H_i^{-1}=H_{n-1-i}$,donde i:0,..,n-1 y n= longitud de los filtros.

L^{-1}

1	1

H^{-1}

-1	1

LL^{-1}

1	1
1	1

LH^{-1}

-1	1
-1	1

HL^{-1}

-1	-1
1	1

HH^{-1}

1	-1
-1	1

B- Reconstrucción

B1- Filtrado de los nodos:

Se inicia desde el último nivel de descomposición integrado por las cuatro matrices generadas por los filtros LL-LH-HL-HH. A partir de éstas, se aplica a cada una el correspondiente filtro HH^{-1},HL^{-1},LH^{-1}, y LL^{-1} y se suma elemento a elemento para obtener la matriz generada por el filtro LL del nivel anterior. Así sucesivamente hasta obtener la raíz, que es la imagen original.

B.2- Aplicación de un filtro inverso:

Se apoya la matriz que representa al filtro sobre una parte del nodo cada vez, realizando nuevamente un producto por convolución entre la imagen y el filtro. Como en la descomposición se decimaba la aplicar el filtro ahora hay que interpolar para recuperar el tamaño original. La interpolación consiste en considerar nulos los valores de los elementos intermedios de las matrices, lo cual se implementa salteando un coeficiente del filtro en ambos sentidos al efectuar el producto por convolución, representando la multiplicación por cero.

f_{22}	f_{20}	m_{03}	m_{04}	m_{05}	...	...			m_{0n}
f_{02}	f_{00}	m_{13}	m_{14}	...	...				...
m_{20}	m_{21}	m_{23}	...	...					
m_{n0}	m_{n1}	m_{n2}	...						m_{nn}

Ejemplo:

-1	-1
1	1

filtro

3	2
4	3

imagen

-3	-2	-2	-3
3	2	2	3
-4	-3	-3	-4
4	3	3	4

resultado

4- OTROS ALGORITMOS DESARROLLADOS Y CONCLUSIONES PARCIALES

- Algoritmos intermedios para determinar la aplicación de los filtros de Daubechies.

  Se hicieron pruebas en una planilla de cálculos, simulando la aplicación de distintos filtros a imágenes (matriz) para determinar cómo debían aplicarse las máscaras de filtrado en la descomposición y reconstrucción.

- Se elaboró un algoritmo que almacena los resultados intermedios de aplicar los filtros en archivos separados con formato RAW para ilustrar la documentación presentada.

- Inicialmente la intención era permitir seleccionar la codificación entre Huffman estático y dinámico. Para ello se elaboró un algoritmo para crear una tabla estática, promediando varias imágenes. A medida que se aplicaba la compresión de una imagen, la tabla era 'alimentada'.

  Se determinó que el algoritmo no era eficiente en cuanto a compresión ya que comprimía mucho menos que el Huffman dinámico, por lo tanto no fue incorporado al algoritmo final.

- El algoritmo fue diseñado para imágenes con formato RAW ; para poder visualizarlas en la interfaz creada, se elaboró un algoritmo de transformación a formato BMP.

- Se desarrolló un algoritmo que calcula la diferencia entre dos imágenes (original y reconstruida) creando una imagen diferencia para ser ilustrada en la interfaz y en la documentación.

- Se desarrolló un algoritmo que calcula el PSNR(relación señal ruido pico) entre dos imágenes (original y reconstruida) para medir la calidad y ser visualizado en la interfaz y en la documentación.

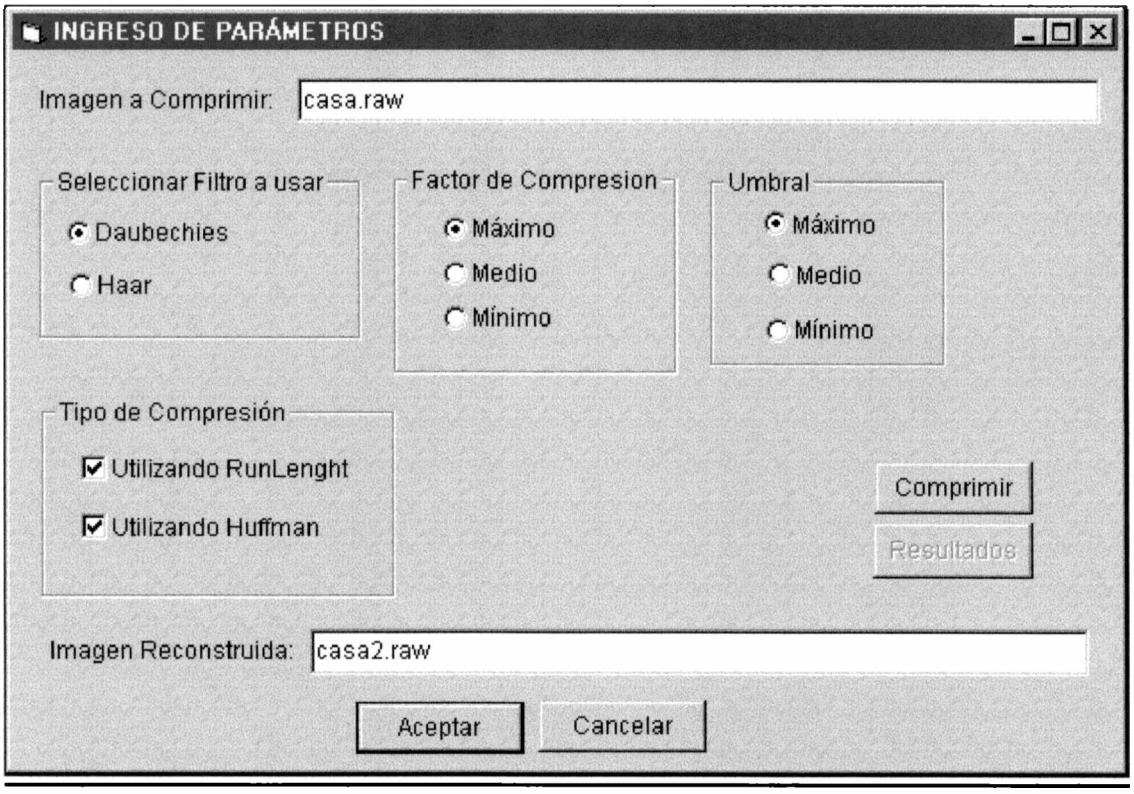
CAPÍTULO IV

INTERFAZ

La interfaz fue elaborada en Visual Basic 4.0.

A continuación mostraremos algunas pantallas de la interfaz.

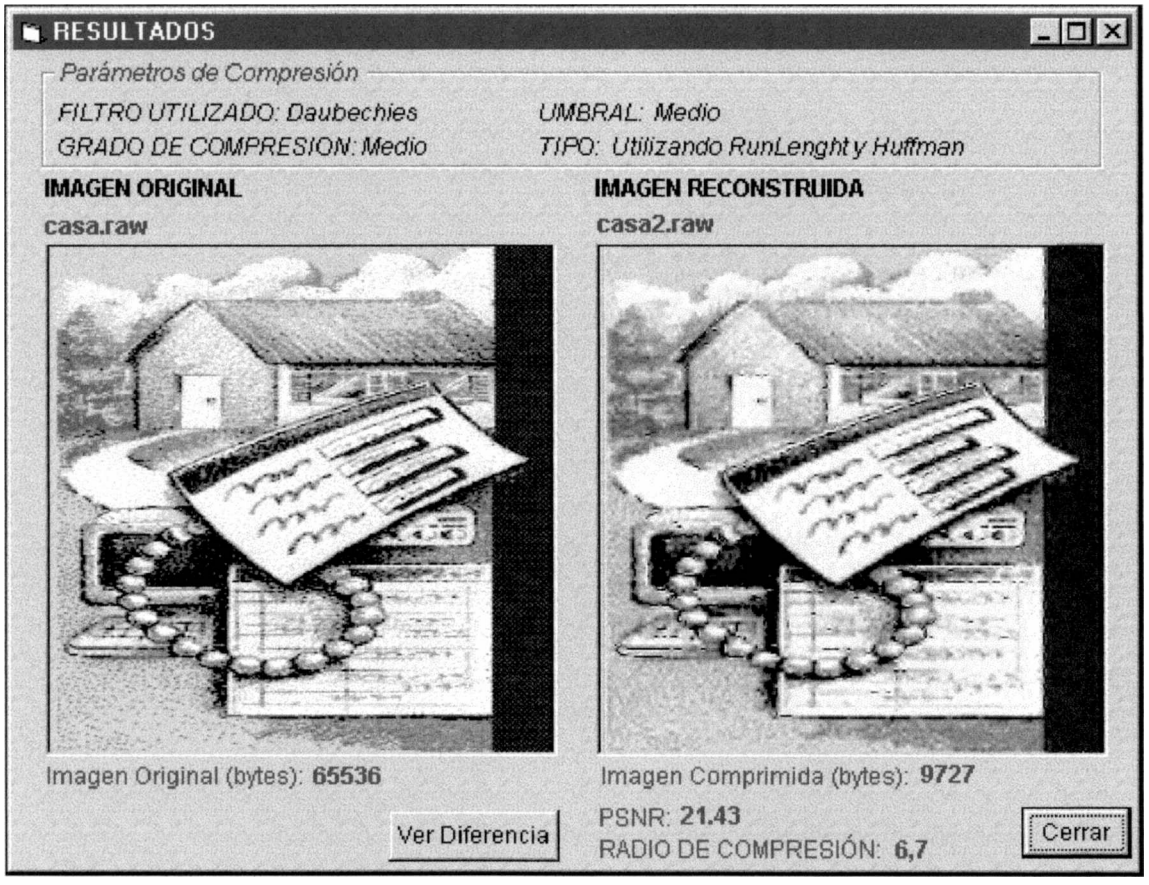
- ***Pantalla de Ingreso de parámetros :***



En esta pantalla se seleccionan las condiciones de compresión :

- Selección del filtro a usar.
- Factor de compresión.
- Umbral : De acuerdo a la opción seleccionada, se establece un número cercano a 0. Los valores menores a ese número(umbral) se sertean a 0.
- Posibilidad de aplicar o no Huffman y Run-Length.

- ***Pantalla de Resultados :***



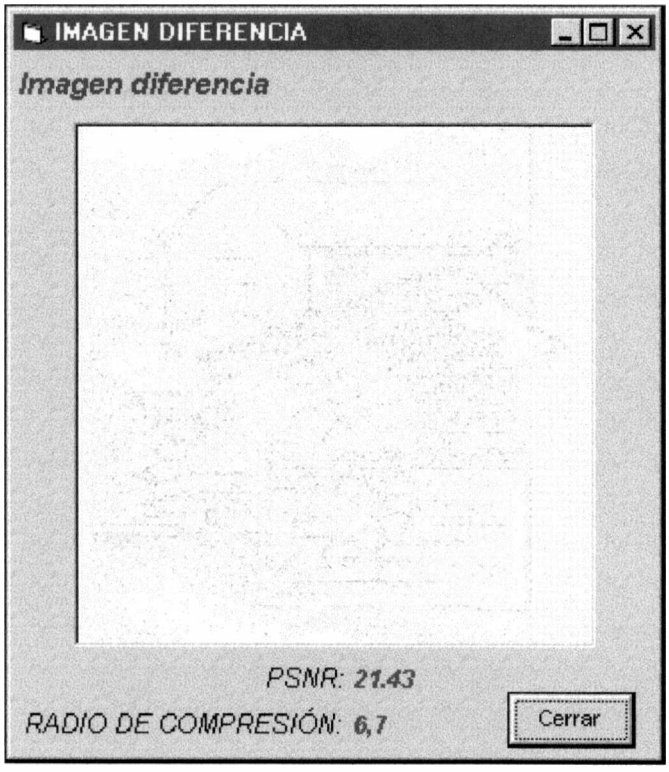
En esta pantalla se muestran los resultados de la compresión.

En la parte superior, se detallan las condiciones de compresión seleccionadas en la pantalla anteriormente descrita.

Luego se muestra la imagen original y reconstruida, con sus tamaños, el cálculo del PSNR y el radio de compresión alcanzado.

Con el botón 'Ver Diferencia' se muestra la imagen diferencia (resta de las dos imágenes).

- ***Pantalla de Diferencias :***



Muestra la imagen diferencia (resta) de la imagen original y reconstruida.

CAPÍTULO V

RESULTADOS

A continuación mostraremos algunos resultados al aplicar el algoritmo sobre algunas imágenes. Los resultados son presentados en cuadros, figuras y gráficos.

Hemos utilizado tres figuras de prueba que son conocidas en procesamiento de imágenes : LENA, LAND y MOLE.

Filtros de Daubechies. Grado de Compresión : **Máximo.**

IMAGEN	TAMAÑO ORIGINAL (BITS)	UMBRAL	COMPRESIÓN		COMPRESIÓN S/HUFFMAN		COMPRESIÓN S/RUNLENGTH		COMPRESIÓN S/HUFFMAN Y RUNLENGTH		PSNR
			BITS	RADIO	BITS	RADIO	BITS	RADIO	BITS	RADIO	
LENA	65.536	Mínimo	6.150	10,6	7.346	8,9	7.280	9,0	21.141	3,0	27,79
		Medio	6.118	10,7	7.335	8,9	7.240	9,0	21.141	3,0	27,78
		Máximo	5.666	11,5	6.964	9,4	6.718	9,7	21.141	3,0	27,58
MOLE	65.536	Mínimo	4.424	14,8	5.037	13,0	6.142	10,6	23.279	2,8	38,49
		Medio	4.384	14,9	4.990	13,1	6.099	10,7	23.279	2,8	38,47
		Máximo	3.443	19,0	3.671	17,8	5.367	12,2	23.279	2,8	36,94
LAND	65.536	Mínimo	13.426	4,8	14.209	4,6	13.422	4,8	14.597	4,4	16,83
		Medio	13.253	4,9	13.989	4,6	13.224	4,9	14.597	4,4	16,83
		Máximo	12.210	5,3	13.953	4,6	11.838	5,5	14.597	4,4	16,81

Filtros de Daubechies. Grado de Compresión : **Medio.**

IMAGEN	TAMAÑO ORIGINAL (BITS)	UMBRAL	COMPRESIÓN		COMPRESIÓN S/HUFFMAN		COMPRESIÓN S/RUNLENGTH		COMPRESIÓN S/HUFFMAN Y RUNLENGTH		PSNR
			BITS	RADIO	BITS	RADIO	BITS	RADIO	BITS	RADIO	
LENA	65.536	Mínimo	7.793	8,4	9.414	6,9	8.895	7,3	22.549	2,9	28,69
		Medio	7.677	8,5	9.363	6,9	8.717	7,5	22.549	2,9	28,67
		Máximo	6.755	9,7	8.387	7,8	7.746	8,4	22.549	2,9	28,36
MOLE	65.536	Mínimo	5.175	12,6	6.157	10,6	6.951	9,4	25.938	2,5	39,34
		Medio	5.044	12,9	5.983	10,9	6.834	9,5	25.938	2,5	39,30
		Máximo	3.767	17,3	4.106	15,9	5.878	11,1	25.938	2,5	37,49
LAND	65.536	Mínimo	9.464	6,9	12.255	5,3	10.150	6,4	22.021	2,9	18,89
		Medio	9.426	6,9	12.199	5,3	10.046	6,5	22.021	2,9	18,89
		Máximo	8.937	7,3	12.133	5,4	9323	7,0	22.021	2,9	18,85

Filtros de Daubechies. Grado de Compresión : **Mínimo.**

IMAGEN	TAMAÑO ORIGINAL (BITS)	UMBRAL	COMPRESIÓN		COMPRESIÓN S/HUFFMAN		COMPRESIÓN S/RUNLENGTH		COMPRESIÓN S/HUFFMAN Y RUNLENGTH		PSNR
			BITS	RADIO	BITS	RADIO	BITS	RADIO	BITS	RADIO	
LENA	65.536	Mínimo	8.932	7,3	10.676	6,1	10.124	6,4	23.573	2,7	28,77
		Medio	8.713	7,5	10.538	6,2	9.735	6,7	23.573	2,7	28,75
		Máximo	7.442	8,8	9.086	7,2	8.419	7,7	23.573	2,7	28,41
MOLE	65.536	Mínimo	6.935	9,4	8.653	7,5	8.691	7,5	29.523	2,2	40,98
		Medio	6.411	10,2	7.936	8,2	8.197	7,9	29.523	2,2	40,89
		Máximo	4.302	15,2	4.830	13,5	6,644	9,8	29.523	2,2	39,39
LAND	65.536	Mínimo	10.914	6,0	13.339	4,9	11.790	5,5	23.045	2,8	18,92
		Medio	10.851	6,0	13.228	4,9	11.585	5,6	23.045	2,8	18,92
		Máximo	10.110	6,4	13.121	4,9	10.463	6,2	23.045	2,8	18,87

Filtros de Haar. Grado de Compresión : **Medio.**

IMAGEN	TAMAÑO ORIGINAL (BITS)	UMBRAL	COMPRESIÓN		COMPRESIÓN S/HUFFMAN		COMPRESIÓN S/RUNLENGTH		COMPRESIÓN S/HUFFMAN Y RUNLENGTH		PSNR
			BITS	RADIO	BITS	RADIO	BITS	RADIO	BITS	RADIO	
LENA	65.536	Medio	7.105	9,2	8.886	7,3	8.088	8,1	21.942	2,9	26,92
MOLE	65.536	Medio	6.199	10,5	7.378	8,8	7.908	8,2	26.068	2,5	18,75
LAND	65.536	Medio	9.394	6,9	12.397	5,2	9.949	6,5	22.029	2,9	36,92

Compresión utilizando JPEG Base Line

IMAGEN	TAMAÑO ORIGINAL (BITS)	RADIO DE COMPRESIÓN	PSNR
LENA	65.536	8,80	32,37
MOLE	65.536	25,24	38,98
LAND	65.536	4,03	24,85

***Comparaciones* - LENA.RAW**

Factor de Compresión : MÁXIMO

Umbral : MÁXIMO

Original : 65.536 bytes

Comprimida : 5.666 bytes

Radio de Compresión : 11,5

PSNR : 27,58

Comparaciones ***- LENA.RAW***

Factor de Compresión : MEDIO

Umbral : MEDIO

Original : 65.536 bytes

Comprimida : 7.677 bytes

Radio de Compresión : 8,5

PSNR : 28,67

Comparaciones* - *LENA.RAW

Factor de Compresión : MÍNIMO

Umbral : MÍNIMO

Original : 65.536 bytes

Comprimida : 8.932 bytes

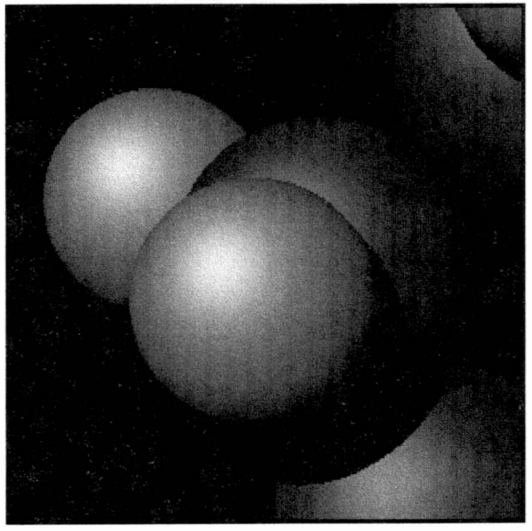
Radio de Compresión : 7,3

PSNR : 28,77

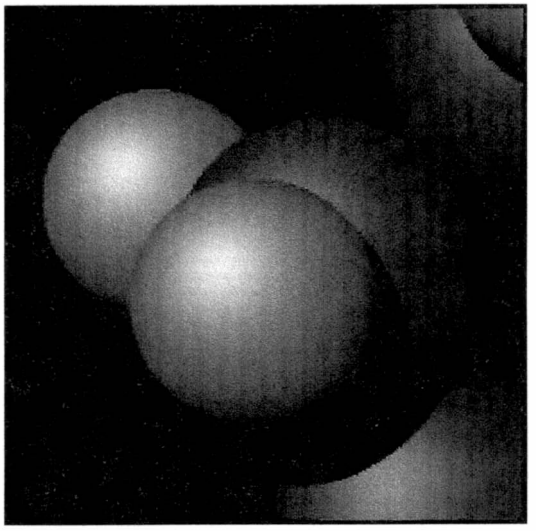
***Comparaciones* - MOLE.RAW**

Factor de Compresión : MÁXIMO

Umbral : MÁXIMO



Original : 65.536 bytes



Comprimida : 3.443 bytes

Radio de Compresión : 19,0

PSNR : 36,94

Comparaciones ***- LAND.RAW***

Factor de Compresión : MEDIO

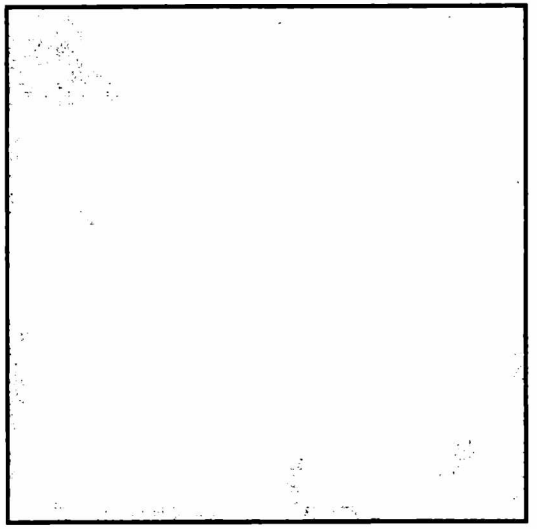
Umbral : MEDIO

Original : 65.536 bytes

Comprimida : 9.426 bytes

Radio de Compresión : 6,9

PSNR : 18,89



***Comparaciones* con JPEG.**

Con el algoritmo desarrollado.

Radio de Compresión : 11,5

PSNR : 27,58

Con JPEG.

Radio de Compresión : 8,8

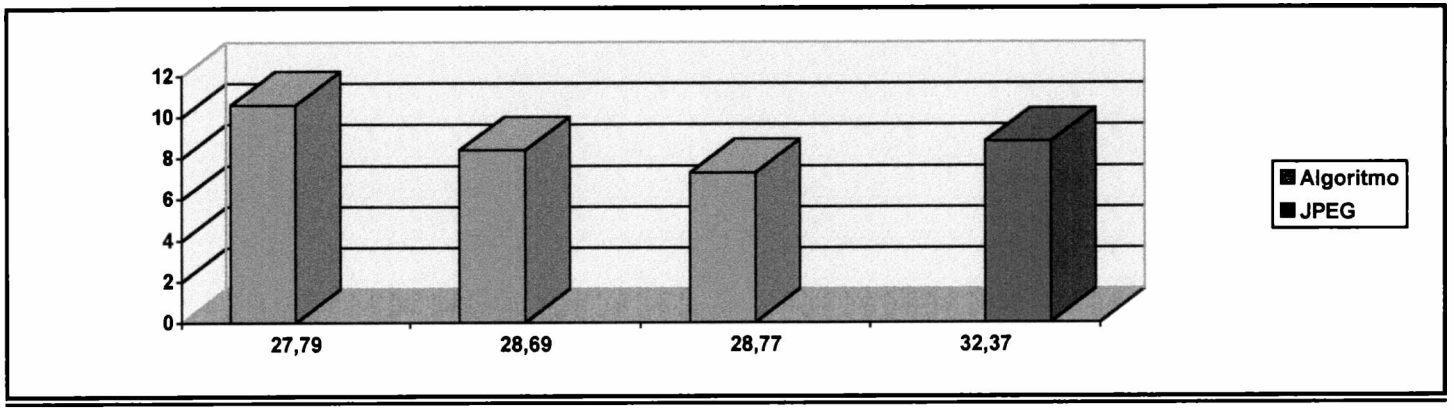
PSNR : 32,37

Con el algoritmo desarrollado.

Radio de Compresión : 8,5

PSNR : 28,67

Comparación con JPEG



CAPÍTULO VI

CONCLUSIONES Y TRABAJOS FUTUROS

1- CONCLUSIONES

Analizando los resultados, concluimos que la aplicación de la transformada wavelet sobre los datos conduce a una pérdida menos perceptible de calidad visual que JPEG cuando se comprime más. Puede agregarse que si bien la cuantificación reduce notablemente el tamaño de la imagen, es la codificación entrópica, llevada a cabo con códigos RunLength y Huffman, la que permite llevar el radio de compresión a valores comparables con JPEG.

La aplicación de la transformada sobre toda la imagen en lugar de por bloques, elimina el problema que tiene JPEG en el cual esos bloques se hacen visibles cuando el radio de compresión usado es muy amplio.

2- TRABAJOS FUTUROS

Como trabajo futuro se sugiere la compresión de imágenes color y video, así como también la aplicación de cuantificación vectorial en lugar de escalar, lo cual es de suponer, incrementará la compresión sin pérdida apreciable de calidad visual.

BIBLIOGRAFÍA

- Alain Fournier, *Wavelets and their Application in Computer Graphics,* SIGGRAPH '94 Course Notes, University of British Columbia.
- Anil K. Jain, *Fundamentals of digital image processing.* Prentice Hall, 1989.
- Tim Edwards, *Discrete Wavelet Transforms: Theory and Implementation*, Stanford University, 1991.
- John A. Robinson, *Binary Tree Predictive Coding*, University of Waterloo, 1993.
- Craig A. Lindley, *JPEG-Like Image Compression*, Dr. Dobb's, 1995.
- *The Fast Wavelet Transform.* Mac. A. Cody, Dr. Dobb's. 1992.
- *The Wavelet Packet Transform.* Mac A. Cody, Dr. Dobb's. 1994.
- John D. Villaseñor, Benjamin Belzer, Judy Liao, *Wavelet Filter Evaluation for Image Compression,* 1995.
- Félix Safar, Fabián Blasetti, Juan Pablo Villa, Daniel Cocimano, Sergio Katz, *Representación adaptiva de imágenes en bases de datos ortogonales óptimas, y aplicaciones a compresión.* Cuarta RPI, Buenos Aires. 1991.
- Trabajo de Grado de la Carrera Licenciatura en Informática de la Facultad de Ciencias Exactas de La Plata. *"Un algoritmo Paralelo Adaptativo de Compresión de Imágenes en tonos de gris".* Juan Pablo Villa.
- Wim Sweldens, Robert Piessens, *Wavelet Sampling Tecniques.*
- R.O. Weels, *Recent Advances in Wavelet Technology,* 1994.
- Stephan G. Mallat, A *Theory for Multiresolution Signal Descomposition. The Wavelet Representation.*
- J.E. Odegard, R.a. Gophinat, C.S. Burrus, *Design of Linear Phase Cosine Modulated Filter Bank for Subband Image Compression.*
- Peter M. Heller, Truong Q. Nguyen, Hemant Singh, W. Knox Carey, *Linear-Phase M-Band Wavelets with Application to Image Coding.*
- Jean-Luc Starck, *Image Restoration with Noise Suppression Using the Wqavelet Transform.* 1994.

- Allen Gersho, Robert M. Gray, *Vector Quantization and Signal Compression.*
- N.S. Jayant, Digital Coding of Waveforms: *Principles and Applications to Speech and Video.* Prentice Hall, 1984
- Devore, R. Jauverth, B.J. Lucier, *Image Compression Through Wavelet Transform Coding.* IEEE Transaction on Information Theori, 1992.
- Alexander D. Poularikas. *The Transforms and applications handbook.* IEE Press.
- R.C.Gonzalez, E. Woods. *Digital Image Processing.* Addison Wesley. 1992.
- Fourier, Castro, Bria, Russo. *Algoritmo de Compresión de Imágenes Fijas utilizando la Transformada Wavelet.* Anales del 3er Congreso Argentino de Ciencias de la Computación (CACIC). La Plata, Argentina. Octubre 1997.
- G.K.Wallance. *The JPEG Still Picture Compression Standard.* Comunications of the ACM, Vol 34. Abril 1991.
- W.K. Pratt. *Digital Image Processing*, John Wiley and Sons, New York, 1991.