

Compresión de imágenes con pérdidas. Método de la Transformada Wavelet

A.I.A. Carolina Celeste Gómez

*Trabajo Práctico Final – Cátedra de Captura y Procesamiento Digital de Señales e Imágenes
Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral*

Introducción

Las técnicas de compresión de imágenes pueden dividirse en dos grandes grupos:

Técnicas de Compresión Sin Pérdidas o Libre de Errores: dentro de este primer grupo, las más conocidas son la codificación de Huffman, RLC o Código de Longitud Variable y LZW. Este tipo de métodos se emplea en aquellos casos en los cuales, ya sea por razones legales o por el costo del proceso de adquisición de los datos, no es admisible la pérdida de información. Las tasas de compresión que se alcanzan son normalmente de 2 a 10. Generalmente involucran dos operaciones: a) mapeo o representación alternativa de la información a fin de eliminar la redundancia interpixel; b) codificación de la información, para eliminar la redundancia de código.

Técnicas de Compresión con Pérdidas: las mismas, se basan en comprometer la fidelidad en la reconstrucción de la imagen a cambio de un incremento en la tasa de compresión, logrando relaciones de hasta 100:1 en imágenes monocromas. La principal diferencia que presentan con respecto a los métodos indicados anteriormente es que el proceso involucra cuantización. Es precisamente en esta operación que se “pierde” la información, ya que la cuantización es un proceso irreversible. El más popular de estos métodos es el denominado JPEG.

En los últimos años se han invertido muchos esfuerzos en el estudio de nuevas técnicas para el procesamiento de señales e imágenes. Estas técnicas se basan en el análisis multiresolución, es decir, representan y estudian los datos en más de una resolución, por ejemplo tiempo-frecuencia. Dentro de los análisis multiresolución se destaca la Transformada Wavelet, debido a su habilidad para: efectuar análisis locales, detección de bordes y discontinuidades, entre otras.

El presente trabajo se ocupa de investigar el uso de esta transformación como una nueva alternativa en la tarea de comprimir imágenes con pérdida de información.

1. Transformada Wavelet

Las wavelets son familias de funciones que tienen la particularidad de tener soporte acotado, es decir son acotadas en el tiempo, una media igual a cero. Cada familia está compuesta por un conjunto de wavelets que son versiones trasladadas y escaladas de una wavelet madre.

En el caso de dos dimensiones, se tiene: una función de escalado, separable, $\Phi(x,y) = \Phi(x) \Phi(y)$ y tras wavelets “sensitivas direccionalmente”, que miden las variaciones de intensidad o de grises en las distintas direcciones:

$$\Psi^H(x,y) = \Psi(x) \Phi(y) \quad \text{dirección horizontal (columnas),}$$

$$\Psi^V(x,y) = \Phi(x) \Psi(y) \quad \text{dirección vertical (filas),}$$

$$\Psi^D(x,y) = \Psi(x) \Psi(y) \quad \text{dirección diagonal.}$$

Estas funciones se definen como:

$$\Phi_{j,m,n}(x,y) = 2^{j/2} \Phi(2^j x - m, 2^j y - n)$$

$$\Psi^i_{j,m,n}(x,y) = 2^{j/2} \Psi^i(2^j x - m, 2^j y - n), \quad \text{donde } i = H, V, D.$$

La Transformada Wavelet Discreta de una función $f(x,y)$ de $M \times N$, está dada por:

$$W_j(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) j_{j_0, m, n}(x, y)$$

$$W^i_\Psi(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \Psi^i_{j, m, n}(x, y)$$

donde j_0 es una escala inicial arbitraria y los coeficientes $W_j(j_0, m, n)$ definen una aproximación de $f(x, y)$ a la escala j_0 . Los coeficientes $W^i_\Psi(j_0, m, n)$ corresponden a los detalles en las tres direcciones para las escalas $j \geq j_0$.

La transformación inversa está dada por:

$$f(x, y) = \frac{1}{\sqrt{MN}} \left[\sum_m \sum_n W_j(x, y) \Psi^i_{j, m, n}(j_0, m, n) j_{j_0, m, n}(x, y) + \sum_{h, v, d} \sum_{j=j_0}^{\infty} \sum_m \sum_n W^i_\Psi(j, m, n) Y^i_{j, m, n}(x, y) \right]$$

En 1988, Mallat desarrolló un algoritmo conocido como Fast Wavelet Transform que consiste en una optimización en el cálculo de la DWT. Este algoritmo puede emplearse en aquellos casos en que la función wavelet posea su correspondiente función de escalado.

2. Codificación Wavelet. Descomposición Multinivel y Reconstrucción.

La codificación wavelet se basa en la idea de que los coeficientes de una transformación que decorrelaciona los píxeles de una imagen pueden codificarse en forma más eficiente que los píxeles originales.

Si la wavelet utilizada como base puede concentrar la mayoría de la información visualmente importante en unos pocos coeficientes, los coeficientes restantes pueden cuantizarse groseramente, o truncarse a cero, con una pequeña distorsión de la imagen.

Este análisis consiste en descomponer una señal o imagen en una serie de aproximaciones y detalles organizados jerárquicamente en niveles.

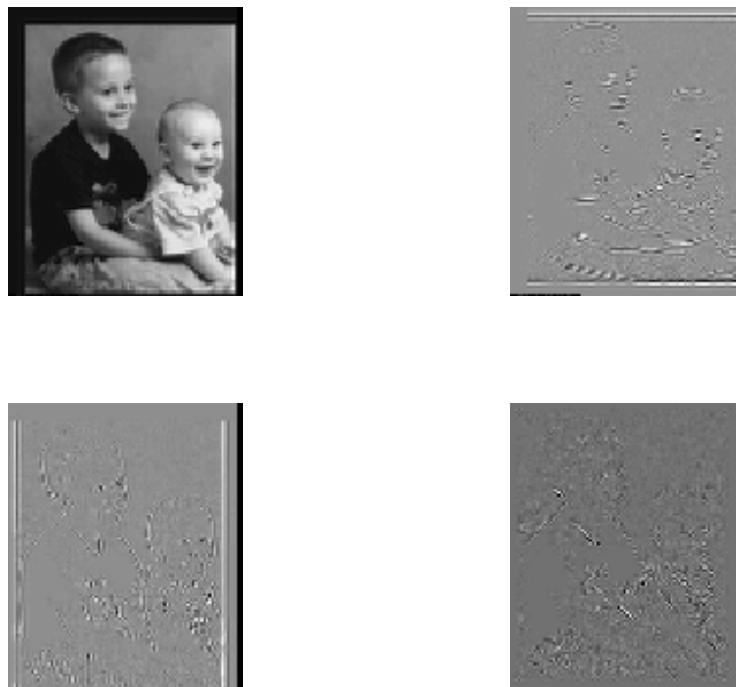


Fig.1 Descomposición de una imagen en 2 niveles.

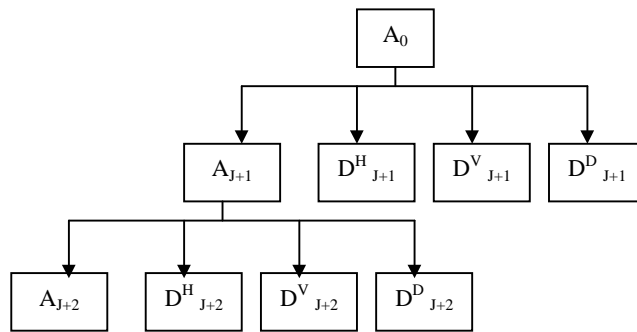


Fig. 2 Descomposición Wavelet en niveles

Para cada nivel j se construye la aproximación A_j , y una serie de detalles D^h , D^v , D^d . La imagen original puede considerarse como la “aproximación de nivel 0”. Se emplean los términos aproximación y detalle debido a que A_j es una aproximación para A_0 obtenida a partir de sus “bajas frecuencias” y los detalles D^i corresponden a las correcciones de “altas frecuencias”.

Una forma más práctica y simple de ver el proceso es suponer que las imágenes A_1 A_2 A_3 son capturadas sucesivamente con el mismo dispositivo, pero disminuyendo la resolución en cada captura. Así, las distintas imágenes son aproximaciones sucesivas, y el detalle es la diferencia entre dos imágenes sucesivas. De esta forma, por ejemplo:

$$A_2 = A_3 + D_3 = A_4 + D_4 + D_3.$$

Para el resultado del proceso de descomposición y reconstrucción sea satisfactorio deben tenerse en cuenta tres cuestiones muy importantes: la función wavelet a utilizar, el nivel de descomposición y la forma de la cuantización.

- *Selección de la función wavelet:* la familia de wavelets elegida afecta tanto al diseño como a la performance del sistema de compresión, y dependerá tanto de la señal a analizar como de la experiencia previa.

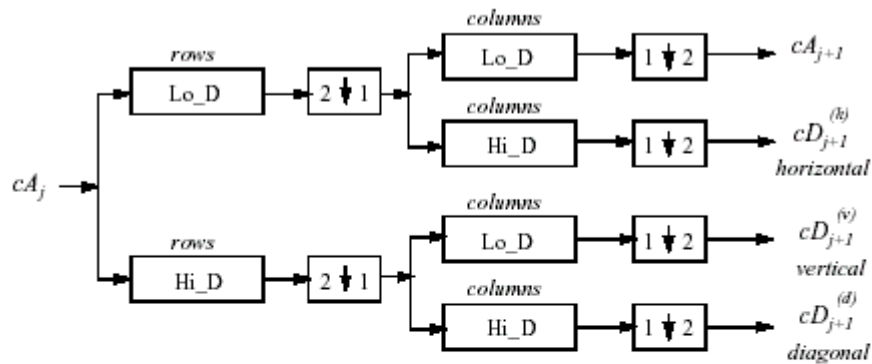
Por ejemplo, cuando la wavelet seleccionada posea una función de escalado, la transformación puede realizarse empleando el algoritmo de la FWT, agilizando muchísimo los cálculos.

Las funciones más utilizadas son Daubechies y Biortogonales, debido a su número de momentos nulos y al suavizado de la reconstrucción.

- *Selección del nivel de descomposición:* el número de operaciones de filtrado depende del nivel de descomposición, por este motivo debe tenerse cuidado en la determinación de este parámetro. Por otro lado, la elección de un nivel demasiado alto puede llevar a inconvenientes en la reconstrucción debido a la cuantización en los detalles.
- *Modo de Cuantización:* es el principal factor que afecta a la tasa de compresión. Puede llevarse a cabo de distintas formas: mediante umbralado en los coeficientes de detalle, o como función de la entropía.

Una vez que se han determinado estos parámetros, la descomposición y reconstrucción se llevan a cabo siguiendo los pasos que se presentan en los diagramas de bloques siguientes:

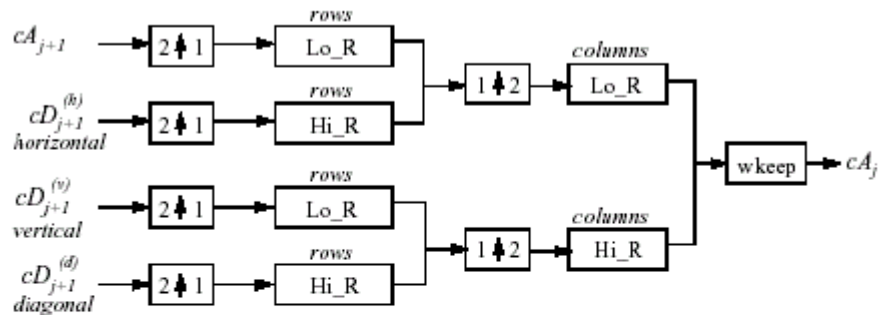
Decomposition step



- Where:
- $\begin{bmatrix} 2 & \downarrow & 1 \end{bmatrix}$ Downsample columns: keep the even indexed columns.
 - $\begin{bmatrix} 1 & \downarrow & 2 \end{bmatrix}$ Downsample rows: keep the even indexed rows.
 - $\begin{matrix} \text{rows} \\ \boxed{X} \end{matrix}$ Convolve with filter X the rows of the entry.
 - $\begin{matrix} \text{columns} \\ \boxed{X} \end{matrix}$ Convolve with filter X the columns of the entry.

Initialization $CA_0 = s$ for the decomposition initialization.

Reconstruction step



- Where:
- $\begin{bmatrix} 2 & \uparrow & 1 \end{bmatrix}$ Upsample columns: insert zeros at odd-indexed columns.
 - $\begin{bmatrix} 1 & \uparrow & 2 \end{bmatrix}$ Upsample rows: insert zeros at odd-indexed rows.
 - $\begin{matrix} \text{rows} \\ \boxed{X} \end{matrix}$ Convolve with filter X the rows of the entry.
 - $\begin{matrix} \text{columns} \\ \boxed{X} \end{matrix}$ Convolve with filter X the columns of the entry.

3. Codificación Wavelet para compresión de imágenes: caso práctico.

Para probar el desempeño de la transformada wavelet en la compresión de imágenes se seleccionó una imagen de 400x318 píxeles en escala de grises con gran nivel de detalles, y se desarrollaron funciones en Matlab para efectuar la descomposición y reconstrucción de la imagen.

Se establecieron los siguientes parámetros para el análisis:

Función wavelet: db6.

Nivel de descomposición : los algoritmos se probaron con tres niveles distintos 8., 90 y 180.

Cuantización: umbralado dependiente del nivel de descomposición. Es decir, en cada nivel se toma la mediana del valor absoluto de los coeficientes del detalle horizontal y luego se llevó a cero a todos los coeficientes de los detalles cuyo valor absoluto estaba por debajo de ese valor.



Fig. 3 Imagen Original

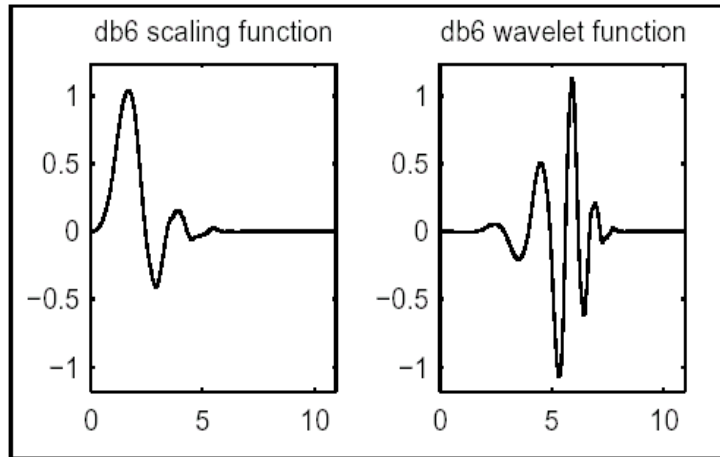


Fig. 4 Función Wavelet seleccionada

Una vez determinado esto, los cálculos se realizaron según el algoritmo de la FWT, aplicando los filtros ortogonales pasa-bajos y pasa-altos obtenidos para la wavelet seleccionada.

Los coeficientes de los detalles obtenidos para cada nivel se fueron guardando en un vector en el que, por último, se colocó la aproximación del último nivel.

Por otro lado, en otra matriz, se fueron guardando los tamaños (filas, columnas) correspondientes a los distintos detalles y aproximaciones. A partir de esta matriz es que luego pueden recuperarse los distintos coeficientes para efectuar el proceso de reconstrucción de la imagen.

Una vez finalizada la descomposición puede aplicarse cualquier algoritmo de compresión de código por ejemplo Huffman o RLC, sobre el vector que contiene los coeficientes para lograr un archivo de menor tamaño. Debido a la gran cantidad de ceros presentes en dicho vector, generados al umbralar los coeficientes de detalle, se logran tasas de compresión bastante altas, sin disminuir por esto la calidad de la imagen reconstruida.

Al obtener la reconstrucción de la imagen, aplicando el algoritmo de la FWT Inversa, y visualizando la aproximación así obtenida junto a la imagen original se observa que, al menos aparentemente, las dos imágenes son iguales.

Para verificar si esto es correcto, se calculó el Error Cuadrático Medio entre ambas imágenes y la energía de las mismas, obteniéndose los siguientes resultados:

nivel = 8 ECM entre la Imagen Original y la Imagen Reconstruida: ans = 2.5365e-005 Porcentaje de compresión logrado: porc_ceros = 50.9234 Energía de la Imagen Original: en_img = 2.4488e+004 Energía de la Imagen Reconstruida: en_img = 2.4488e+004	nivel = 90 ECM entre la Imagen Original y la Imagen Reconstruida: ans = 2.5365e-005 Porcentaje de compresión logrado: porc_ceros = 50.1803 Energía de la Imagen Original: en_img = 2.4488e+004 Energía de la Imagen Reconstruida: en_img = 2.4488e+004	nivel = 150 ECM entre la Imagen Original y la Imagen Reconstruida: 2.5365e-005 Porcentaje de compresión logrado: porc_ceros = 49.7616 Energía de la Imagen Original: en_img = 2.4488e+004 Energía de la Imagen Reconstruida: en_img = 2.4488e+004
---	--	---

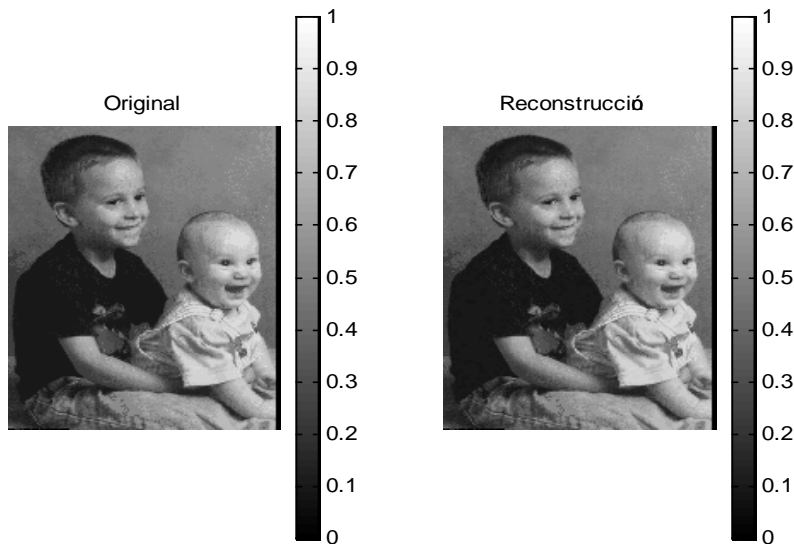


Fig. 5 Resultado del proceso de reconstrucción wavelet

Comprobamos de esta forma que la reconstrucción de la imagen se logra con un margen de error muy pequeño, prácticamente imperceptible para el sistema de la visión humana. Lográndose distintas tasas de compresión, según el nivel de descomposición seleccionado. Puede observarse que llega un punto en el cual aumentar el nivel de descomposición no significa aumentar la tasa de compresión. Por ejemplo, cuando nivel = 180 la reconstrucción de la imagen reconstruida es buena, pero el porcentaje de compresión disminuye.

4. Conclusiones.

La transformada wavelet es un método muy efectivo para abordar el análisis de señales e imágenes.

Cuando se seleccionan en forma correcta la función wavelet a utilizar y el nivel de descomposición, los cálculos se dan en forma rápida y los costos computacionales son bastante bajos.

Por otro lado, si la forma de cuantización se define cuidadosamente, el proceso logrará colocar una gran cantidad de ceros en los coeficientes de los detalles de las descomposiciones en todos los niveles, permitiendo obtener altas tasas de compresión.

La característica más importante que cabe destacar es que, si bien en el proceso está involucrada la cuantización produciéndose como consecuencia la pérdida de información, la imagen logra reconstruirse con muy poca pérdida de detalle.

Este proceso es aplicable tanto a imágenes monocromas como a imágenes color. En este último caso, el proceso de descomposición debe realizarse sobre cada plano de color RGB, teniendo cuidado de que el mapa de colores esté suavizado. En el análisis de este tipo de imágenes, el número de cálculos es mayor y se obtienen tasas de compresión menores, pero los resultados son igual de favorables que en el caso de imágenes en escala de grises.

5. Referencias.

- Wavelet Toolbox for use with Matlab. *Michel Misiti, Yves Misiti, Georges Oppenheim, Jean-Michel Poggi*. 1996 - 1997 by The MathWorks, Inc. All Rights Reserved.
- Digital Image Processing. Gonzalez-Woods.
- Apuntes de la Cátedra de Captura y Procesamiento Digital de Señales e Imágenes, Facultad de Ing. Y Cs. Hídircas, Universidad Nacional del Litoral. Año 2005.

Anexo A: Algoritmos

```

%*****
%PROGRAMA PRINCIPAL: ANÁLISIS WAVELET - DESCOMPOSICIÓN Y RECONSTRUCCIÓN
%*****
clc;    clear all;  close all;

%*****
%Seteo de las preferencias de visualización
%*****
iptsetpref('ImshowBorder','tight');
iptsetpref('ImshowAxesVisible','off');
iptsetpref('ImshowTruesize','auto'); %'manual');
iptsetpref('TruesizeWarning','off');

%*****
%Carga de la imagen a comprimir
%*****
[img, map] = imread('kids.tif');    img = ind2gray(img , map);

subplot(1,2,1);    imshow(img);    title('Original');    colorbar;

%*****
%Obtención del mínimo nivel posible de compresión
%*****
nivel = max(floor(log2(size(img))))
wavelet = 'db6';

%*****
%
%           Descomposición Wavelet
%*****
[wcoef, pos] = descomposicion(img, nivel, wavelet);
save wcoef wcoef
save pos pos

%*****
%
%           Porcentaje de compresión logrado
%(calculado en base a la cantidad de 0s del vector de descomposición)
%*****
n = length(wcoef);
n_ceros = length(find(wcoef==0));
porc_ceros = 100 * (n_ceros/n);

%*****
%
%           Reconstrucción Wavelet
%*****
img_rec = reconstruccion(wcoef, pos, wavelet);
subplot(1,2,2);    imshow(img_rec);    title('Reconstrucción');    colorbar;

%*****
%
%           Resultados
%*****
disp('ECM entre la Imagen Original y la Imagen Reconstruída: ');
f_mse(im2double(img),img_rec)

disp(' ');

disp('Energía de la Imagen Original: ');
en_img = norm(im2double(img),2).^2

disp(' ');

disp('Energía de la Imagen Reconstruída: ');
en_img = norm(img_rec,2).^2

```

```

%*****
%*****
function [wcoef, pos] = descomposicion_wavelet(img, nivel, wavelet),
%*****
%*****

%Construcción de los filtros ortogonales
load db6; wavelet = db6;
[lpd, hpd, lpr, hpr] = orthfilt(wavelet);
clear db6 lpr hpr %borra los filtros de reconstruccion que no se van a usar

%Descomposición en niveles
cA = img; cH = []; cV = []; cD = [];

%Inicializacion del vector de posiciones y del vector de coeficientes
pos = [];
pos = [size(cA)];
wcoef = [];

for j = 1:nivel,
    aux_cA = cA;

    auxi = dyaddown(conv2(aux_cA,lpd),'c'); %submuestreo de columnas

    cA = dyaddown(conv2(auxi,lpd'),'r');%submuestreo de filas
    cH = dyaddown(conv2(auxi,hpd'),'r');%submuestreo de filas

    auxi = dyaddown(conv2(aux_cA,hpd),'c'); %submuestreo de columnas

    cV = dyaddown(conv2(auxi,lpd'),'r');%submuestreo de filas
    cD = dyaddown(conv2(auxi,hpd'),'r');%submuestreo de filas

    %Umbralado
    %umbral = median(sort(cH(:)))
    umbral = median(sort(abs(cH(:)))));

    cH(find(abs(cH)<umbral)) = 0;
    %length(find(cH==0))

    cV(find(abs(cV)<umbral)) = 0;
    %length(find(cV==0))

    cD(find(abs(cD)<umbral)) = 0;
    %length(find(cD==0))

    pos = [size(cH); pos];

    wcoef = [cH(:);cV(:);cD(:);wcoef ];

end;

pos = [size(cA); pos];
wcoef = [cA(:);wcoef ]';

```

```

%*****
%*****
function [cA] = reconstruccion(wcoef, pos, wavelet),
%*****
%*****

%Construcción de los filtros ortogonales
load db6;
wavelet = db6; clear db6;

[lpd, hpd, lpr, hpr] = orthfilt(wavelet);

%*****
%                               Reconstrucción
%*****

[row, col] = size(pos);
nivel = row - 2;

inicio = 1 ;   fin = prod(pos(1,:));
cA = reshape(wcoef(inicio:fin), pos(1,1), pos(1,2)) ;

pos = pos(2:row,:);

for n = 1:nivel,
    tamaño = prod(pos(n,:));

    inicio = fin + 1;           fin = inicio + tamaño - 1;
    cH = reshape(wcoef(inicio:fin), pos(n,1), pos(n,2)) ;

    inicio = fin + 1;           fin = inicio + tamaño - 1;
    cV = reshape(wcoef(inicio:fin), pos(n,1), pos(n,2)) ;

    inicio = fin + 1;           fin = inicio + tamaño - 1;
    cD = reshape(wcoef(inicio:fin), pos(n,1), pos(n,2)) ;

    t0 = conv2(dyadup(cA,'r'),lpr') + conv2(dyadup(cH,'r'),hpr');
    d0 = conv2(dyadup(cV,'r'),lpr') + conv2(dyadup(cD,'r'),hpr');

    cA = conv2(dyadup(t0,'c'),lpr) + conv2(dyadup(d0,'c'),hpr);

    cA = wkeep(cA,pos(n+1,:));

end;

return;

```